MBC-770 SERIES
USER'S GUIDE

SANYO

FCC NOTICE

This equipment generates and uses radio frequency energy and if not installed and used properly, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient the receiving antenna.
Relocate the computer with respect to the receiver.
Move the computer away from the receiver.
Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet by the Federal Communications Commission helpful:

"How to Identify and Resolve Radio-TV Interference Problems".

This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock No. 004-000-00345-4.

All rights reserved. No part of this manual may be reproduced in any form without express written permission from Sanyo.

The contents of this manual are subject to change. If you wish to receive revisions and updates, please contact your Sanyo dealer.

Sanyo will not assume responsibility for errors in this manual or their consequences.

MS-DOS is a trademark of Microsoft Corporation.

PREFACE

This manual is divided into eight chapters that are organized in the following manner. Plese read this manual throughtly before you begin to install and use the Sanyo MBC-770 series computer.

Chapter 1. "Getting Started," contains all the information that you will need in order to begin operating your computer such setup and how to use disks.

Chapter 2. "Sanyo GW-BASIC," contains information that you need to know before programming in BASIC and explains the symbols used in Chapter 3 and how to use the Editor.

Chapter 3. "BASIC commands, functions, and statements," is a detailed reference for using Sanyo GW-BASIC.

Chapter 4. "MS-DOS instruction", is an introduction to using the MS-DOS operating system.

Chapter 5. "MS-DOS commands," is for reference when using MS-DOS.

Chapter 6. "Technical reference," is a collection of information needed for advanced programming. You'll find the specification of this macine at the beginning of this chapter.

Chapter 7. "Optional peripheral installation," contains information needed to connect other devices to the computer.

Chapter 8. "Computer vocabulary," is a glossary of computer terms that may be unfamiliar to newcomers.

We suggest that you read Chapter 1 before beginning to use your computer. Refer to Chapter 8 for definitions of unfamiliar terms.

To program the computer using the BASIC language, begin by reading Chapter 3 and refer to Chapter 4.

To use the advanced functions of MS-DOS, start by reading Chapter 5.

CONTENTS

CHAPTER 1. GETTING STARTED

Introduction 1-2
Setting Up 1-3
Unpacking 1-3
Locating the computer 1-3
Setting up the computer 1-4
Handling your disks 1-8
Recommended disks 1-9
Names and Functions of the Parts of the Computer 1-10
Front parts and their functions 1-10
Rear parts and their functions 1-12
Operations 1-14
Disk drive 1-14
Keyboard 1-16
Running the operating system 1-23
Formatting your disks 1-26
How to copy your files 1-28
Resetting the system 1-31
Turning the system power off 1-31

CHAPTER 2. SANYO GW-BASIC

Introduction 2-2
Overview 2-2
Syntax notation 2-2
Using the BASIC Interpreter 2-4
Starting BASIC 2-4
Modes of operation 2-4
Line format 2-5
Learning the Language 2-6
Character set 2-6
Constants 2-7
Variables 2-8
Expressions and operators 2-10
Type conversion 2-17
Functions 2-18
Writing Programs Using the Screen Editor 2-19
Full screen editor 2-19

Working with Files 2-23
Default drive 2-23
Filenames and paths 2-23
Handling files 2-26

CHAPTER 3. BASIC COMMANDS, FUNCTIONS, AND STATEMENTS

CHAPTER 4. MS-DOS INSTRUCTION

Introduction 4-2
What is MS-DOS? 4-2
MS-DOS files 4-2
Files 4-2
More about Files 4-4
How to name your files 4-4
Wild cards 4-5
Illegal filenames 4-7
Directories 4-7
Filenames and paths 4-8
Learning about Commands 4-10
Types of MS-DOS commands 4-10
Information common to all MS-DOS commands 4-10
Batch processing 4-11
AUTOEXEC.BAT file 4-12
Input and output 4-12

CHAPTER 5. MS-DOS COMMANDS

MS-DOS Commands 5-3
Batch Commands (Command Extensions) 5-23
MS-DOS Editing and Function Keys 5-28
Special MS-DOS editing keys 5-29
Control character functions 5-32

CHAPTER 6. TECHNICAL REFERENCE

Specifications 6-2
Character Codes 6-4
Basic Reserved Words 6-7
Basic Error Messages 6-9
Disk Error Messages 6-12
Memory Map 6-15
Interrupt Vectors 6-16
Interrupt Routines 6-18
I/O Map 6-24
Dip Switches 6-25
Expansion Slot Signals 6-26
Connector Assignment 6-27
DOS Editing Keys 6-29

CHAPTER 7. OPTIONAL PERIPHERAL INSTALLATION

Installation Instructions 7-2
Upper Body Removal 7-3
Top Plate Removal 7-4
Additional Software and Manuals 7-5

CHAPTER 8. COMPUTER VOCABULARY

Introduction 8-2
Vocabulary 8-2

CHAPTER 1.
GETTING STARTED

Introduction 1-2
Setting Up 1-3
- Unpacking 1-3
- Locating the computer 1-3
- Setting up the computer 1-4
- Handling your disks 1-8
- Recommended disks 1-9

Names and Functions of the Parts of the Computer 1-10
- Front parts and their functions 1-10
- Rear parts and their functions 1-12

Operations 1-14
- Disk drive 1-14
  - Inserting a disk 1-14
  - Removing a disk 1-14
  - Write protect seal 1-15
    - Read/write disk 1-15
    - Write-protected disk 1-15
- Keyboard 1-16
  - ASCII keyboard layout 1-16
  - Alphanumeric keys 1-17
  - Cursor control keys 1-19
  - Function keys 1-20
  - Multiple key functions 1-22
- Running the operating system 1-23
- Formatting your disks 1-26
  - Formatting a disk with a double drive system 1-26
  - Formatting a disk with a single drive system 1-27
- How to copy your files 1-28
  - COPY command 1-28
  - DISKCOPY command 1-29
    - Making a disk copy with a double drive system 1-29
    - Making a disk copy with a single drive system 1-30
- Resetting the system 1-31
- Turning the system power off 1-31

Introduction

Thank you for buying a Sanyo computer. We designed your computer for maximum price-performance and we sincerely hope that you like it.

We wrote this manual to assist you in beginning to use your computer. So that we could communicate to the largest readership, we decided to write it in a way that the average person with a little knowledge of computers could understand it. Some of the computer terms may be unfamiliar to newcomers and for that purpose we have included a GLOSSARY.

Since the best way to learn how to use your computer is by actual hands-on experience, we suggest that you experiment with commands and statements that you do not understand. Consult the following books, available from your Sanyo dealer, for further information.

GW-BASIC Reference Manual
MS-DOS Reference Manual

Once again, thank you.

Sanyo Electric Co.Ltd.

Setting Up

Unpacking

Unpack the MBC-770 series computer from the shipping carton and save the packing material for possible future use.

Locating the computer

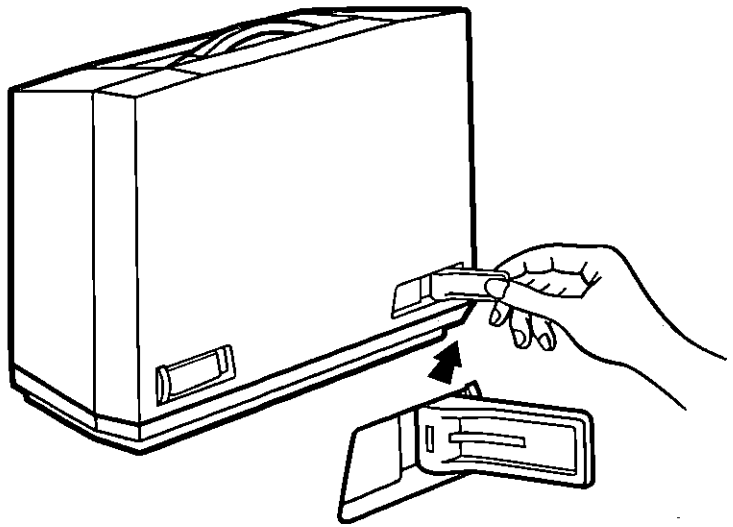
Use the following guidelines for selecting a suitable place for your computer.

1. Do not place the computer where it will be exposed to direct sunlight or sudden changes in temperatures from heaters or coolers.
2. Do not place the computer where it is dirty or dusty, or where vibration may occur.
3. Do not use the computer in an area where there is a strong magnetic field.
4. Place the computer in a well-ventilated room away from walls.
5. Use only specified local power lines.
6. Do not drink or eat while operating the computer.
7. Place the computer in a place where it will not be exposed to spillage.

CAUTION: DO NOT place the printer on top of the computer.

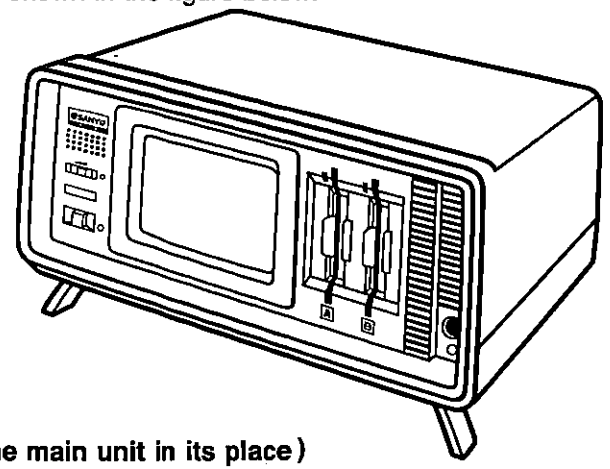
Setting up the computer

Place the computer on a stable surface and open the two hinged feet of the computer as shown in the figure below. These hinged feet raise the computer so that the monitor can be viewed comfortably.



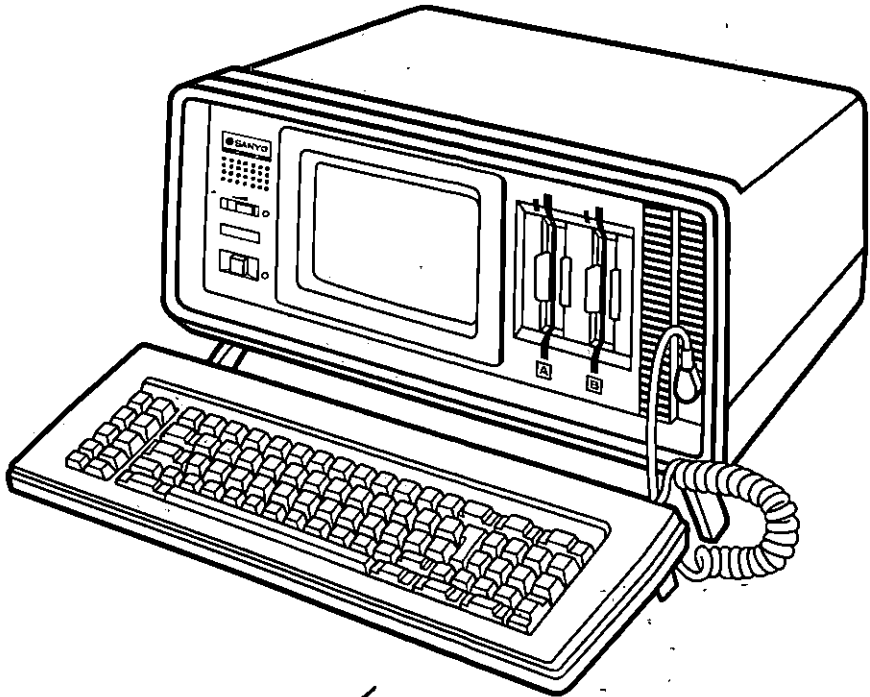
(Opening the feet of the main unit.)

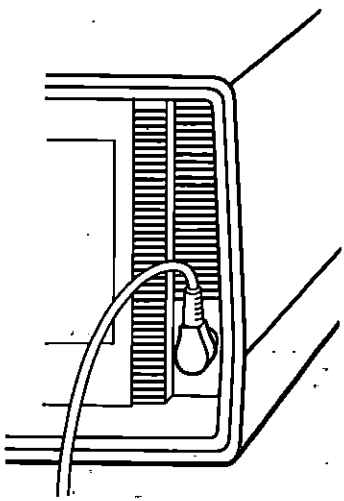
Next place the computer so that it rests on its feet with the handle away from you as shown in the figure below.



(Placing the main unit in its place)

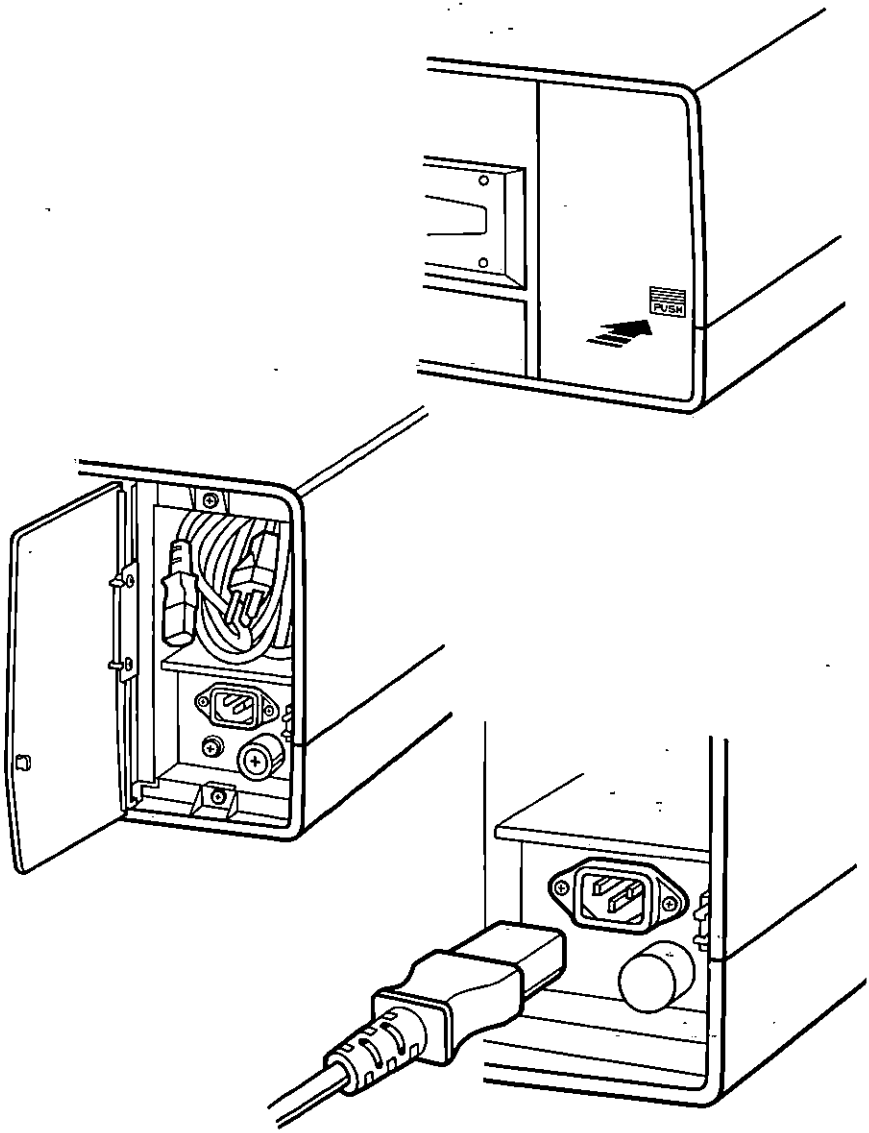
Place the keyboard right-side up on your working surface and plug in the keyboard connector into the place marked "KEYBOARD".





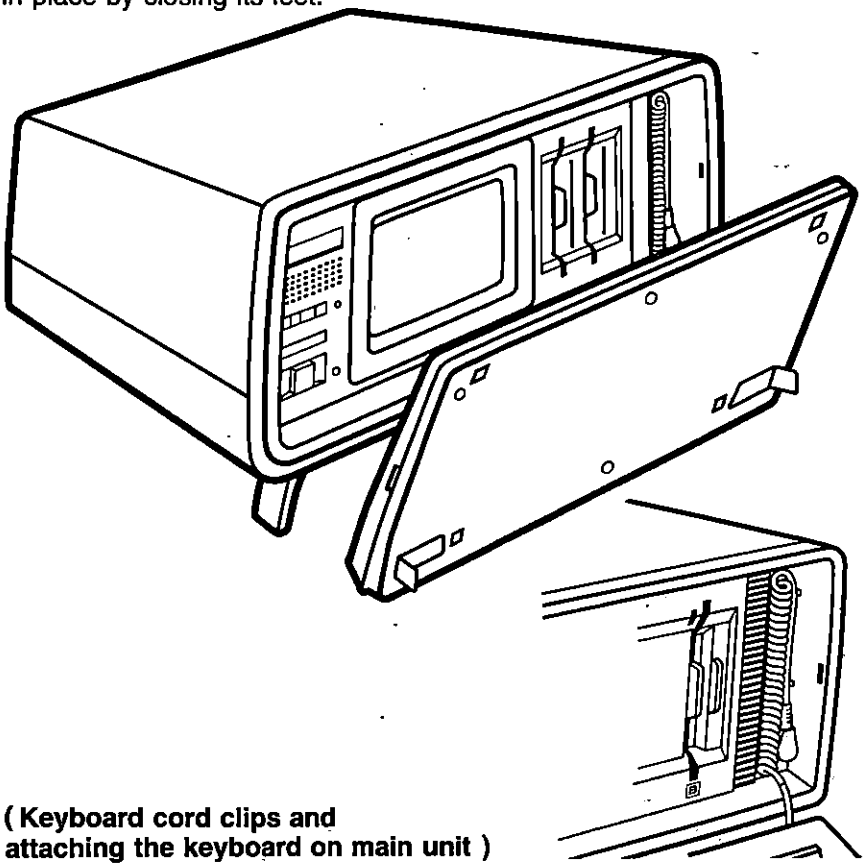
(Connecting the keyboard.)

Connect one end of the AC power cord in the socket illustrated at the right side bottom and the other end in an AC outlet (Use specified local voltage only). The AC connector is located in the connector compartment on the back of the main unit. To open the compartment, press the cover of the compartment. The AC connector can be repacked in the back of the unit and the compartment cover secured by pressing on the top.



(Connector compartment)

When you are done using the computer, you may store away the keyboard on the computer. To attach the keyboard onto the main unit, open its feet and place the keyboard cord on the two clips on the front-right of the main unit. Place the keyboard on the main unit and lock it in place by closing its feet.



(Keyboard cord clips and attaching the keyboard on main unit)

*** * * CAUTION * * ***

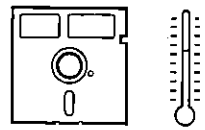
DO NOT ATTACH THE KEYBOARD TO THE MAIN UNIT BEFORE YOU CONFIRM THE POWER IS OFF. THE AIR VENTS MUST BE CLEAR TO COOL THE COMPUTER.

DO NOT BLOCK THESE AIR VENTS WHILE USING THE COMPUTER.

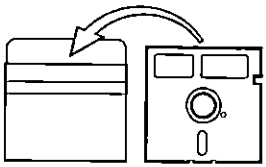
DO NOT PLACE ANY OBJECT LIKE A PRINTER ON THE COMPUTER

Handling your disks

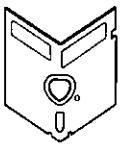
Floppy disks are delicate magnetic media and the data recorded on them is liable to become unreadable if they are handled improperly. Follow the guidelines below in handling disks.



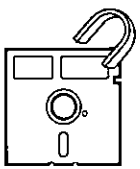
Keep your disks stored away from direct sunlight. Avoid large temperature variations (extreme heat or cold).



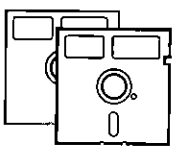
Always store your disks in their envelopes so as to prevent dust or dirt accumulation.



Do not bend or fold your disks.



Keep the disk away from any objects that generate magnetic fields.



Always make copies (backups) of your original disks and those disks with important data to prevent data loss.

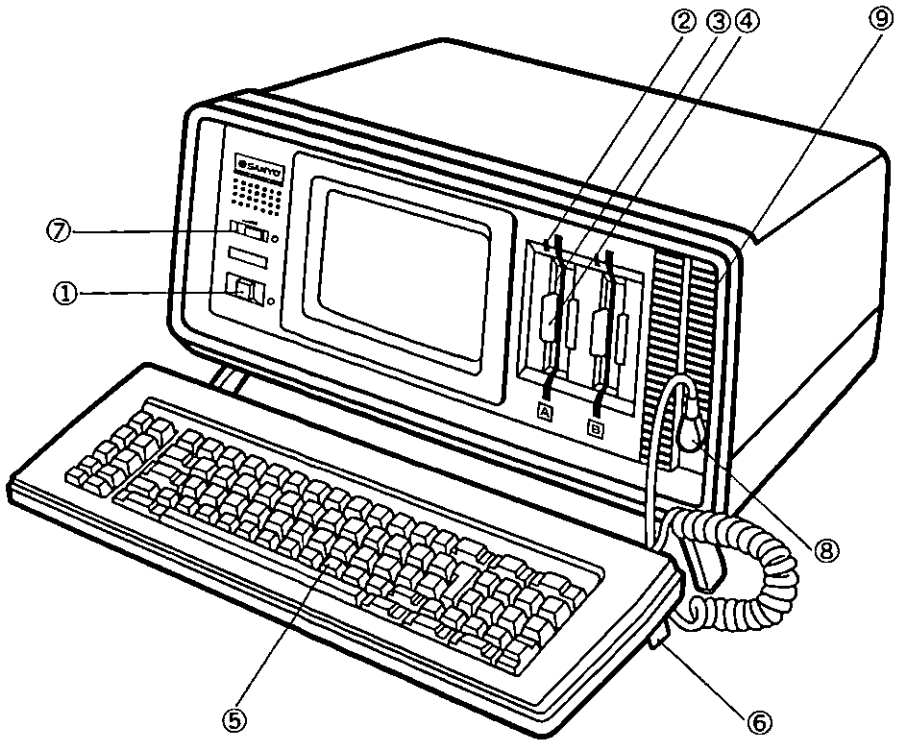
Recommended disks

We recommend that you always use double-sided, double-density, soft-sectored disks. Sanyo will not assume responsibility for damages or other trouble that may arise from the use of other types of disks.

Names and Functions of the Parts of the Computer

Front parts and their functions

1. Power switch
2. Drive lamp
3. Floppy disk drive
4. Disk drive latch
5. Keyboard
6. Tilt leg
7. Display brightness lever
8. Keyboard connector
9. Keyboard cord clips



1. Power switch
Press this switch to turn the computer on and off. A green light will glow above the power switch when the computer is on.

DO NOT turn the power switch on or off with a disk in the drive; this may cause extraneous data to be written on the disk.

2. Drive lamp
A red lamp glows when the disk is being accessed to read or write data.

3. Floppy disk drive(s)
These are built-in 5¼", double-sided, double-density floppy disk drives.

4. Disk drive latch(es)
The disk drive latches are used to secure the floppy in place. These latches must be closed before the drive can read or write data on the disk.

5. Keyboard
The keyboard is an ASCII-type keyboard that contains the normal keys found on any typewriter plus cursor control keys, a numeric keypad, mathematical function keys, editing keys, and programmable function keys.

6. Tilt legs
The keyboard on the Sanyo MBC-770 series computers is equipped with adjustable tilt legs for maximum comfort when typing.

The keyboard may also be used without extending the tilt legs.

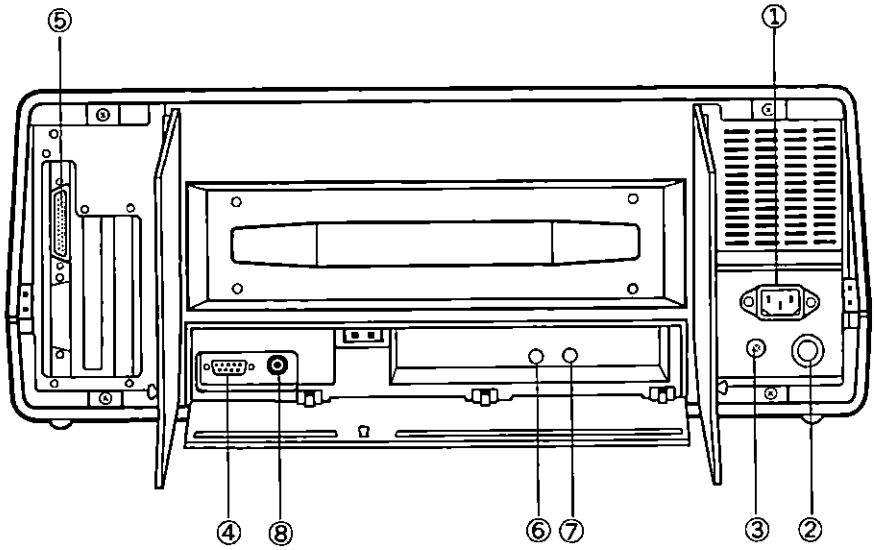
7. Display brightness lever
Move the lever to the left to decrease the brightness; move it to the right to increase the brightness.

8. Keyboard connector
The keyboard connector is used to connect the keyboard to the computer.

9. Keyboard cord clips
These clips are used to hold the cord securely when the computer is stored.

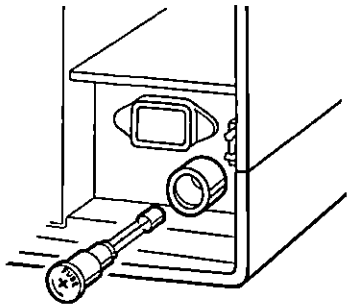
Rear parts and their functions

1. AC connector
2. Fuse
3. Grounding terminal
4. RGB connector
5. Printer connector
6. V-Hold
7. H-Hold
8. Color composite (Video connector)



1. AC connector
Plug one end of the AC power cord in this socket, and the other end in an AC outlet (Use specified local voltage only).

2. Fuse
The computer has a fuse to protect its internal circuits from overheating. Should this fuse blow, turn the power off before replacing it. Use the extra fuse provided or a 1.5 ampere fuse.



3. Grounding terminal
An electrical ground wire may be attached here.

4. RGB connector
This socket is provided to connect high-resolution Color CRT displays to the computer.

5. Printer connector
This connects a parallel Centronics-type printer to the computer.

6. V-Hold
This is used to adjust the vertical hold of the display monitor.

7. H-Hold
This is used to adjust the horizontal hold of the display monitor.

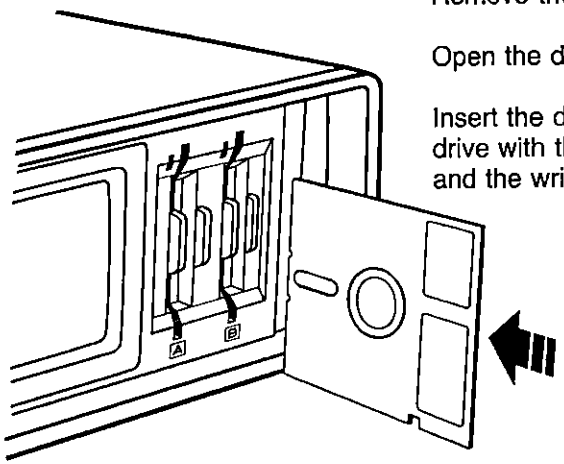
8. Color composite (Video connector)
This outputs a color composite signal for connecting other Color monitors.

Operations

Disk drive

DO NOT insert or remove a disk while the drive is being accessed. Data may be destroyed.

Inserting a disk



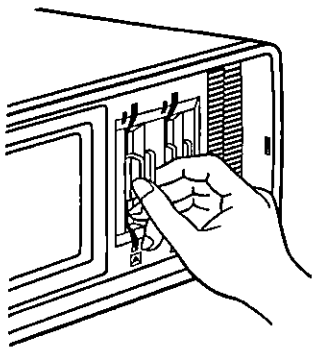
Remove the disk from its envelope.

Open the drive latch.

Insert the disk completely into the drive with the label facing to the left and the write protect notch down.

Close the drive latch.

Removing a disk



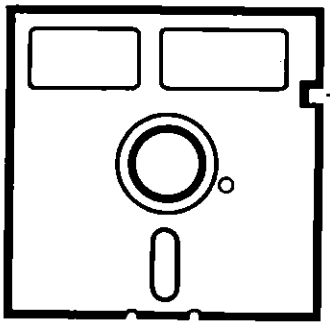
Open the drive by squeezing the drive latch. The disk will pop out. Take the disk out of the drive and place it in its envelope to prevent it from becoming dusty.

Write protect seal

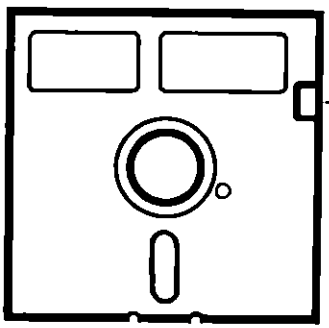
The data written on the disks may be valuable. To protect the data on a disk from being overwritten, cover the notch with a write protect seal. Note that this seal must be removed when you want to write to a disk.

System disks provided with the computer are already write-protected.

Read/write disk



Write-protected disk

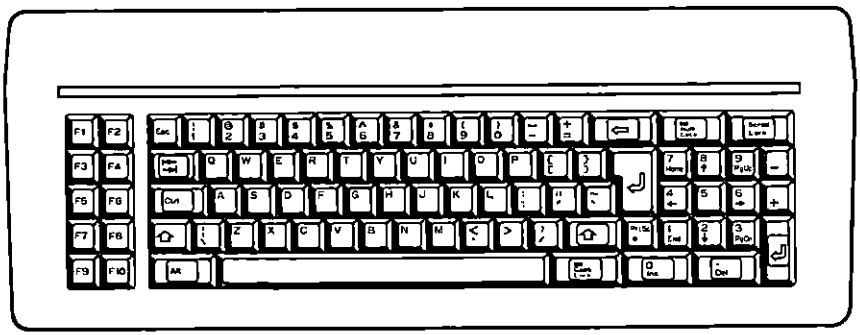


Keyboard

ASCII keyboard layout

The keyboard of the Sanyo MBC-770 series is like a keyboard on a typewriter which has extra functions. When you use a typewriter, the characters are typed onto paper; on the computer the characters appear on the screen. A small blinking underline (called the cursor) shows where the next character will be input.

Note that some of the keys work a little differently from those on a typewriter. For example, while the backspace key causes the cursor to move to the left and erase the character to the left, the Del key causes the computer to erase the character at the cursor position.



Alphanumeric keys

The alphanumeric keys are straightforward in that their use is like that on a typewriter. The number keys are used to enter numbers, the alphabet keys are used to enter letters. Note that the computer also makes a distinction between the letter "l" and the number "1", and the letter "O" and the number "0".

SPACEBAR On a typewriter, the spacebar is used to move the type element to the next position; on the computer, the spacebar is used to enter spaces (blank characters). In the middle of a word, if you press the spacebar, a space replaces the character that was there.

SHIFT

When SHIFT is pressed, the upper case letters are input. With the SHIFT key released, the letters input are lower case.

Caps Lock

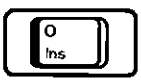
When Caps Lock is pressed, and the shift key is not pressed, upper case letters are input. When the Caps Lock key is pressed and the shift key is pressed, lower case letters are input.

Control keys

Ctrl

This key is used to enter control characters. Hold this key down and then press another key.

Ins



This key is used to toggle between the insert on and off modes. When in the insert on mode, characters are inserted at the cursor. When in the insert off mode, the characters input overwrite any characters that were at the cursor position.

DEL

This key is used to delete characters at the cursor in some applications.

BACKSPACE

This key erases the character to the left of the cursor and moves the cursor back one position.

Esc

The Esc key is used by application programs for various functions.

TAB

The TAB key moves the cursor to the next tab stop in some applications.

ENTER

The ENTER key is used to signal to the computer that a command line is complete.

Num Lock

This key toggles the numeric keypad between number functions and cursor-control functions. When the red lamp is lit, the numeric keypad inputs numbers. When the red lamp is off, the numeric keypad performs cursor-control functions.

Cursor control keys

Home

This key places the cursor at the top left-hand corner of the screen.

Pg Up

This key is used in some applications to scroll the text up.

Pg Dn

This key is used in some applications to scroll the text down.

←

This key moves the cursor to the left.

→ This key moves the cursor to the right.

↓ This key moves the cursor down.

↑ This key moves the cursor up.

Break This key is used in combination with the Ctrl key to break the listing and execution of a BASIC program.

Function keys

The keys labeled F1 to F10 are programmable function keys that can input a previously defined sequence of characters. The definitions assigned to these function keys depend on the application being used. In MS-DOS, the function keys and other keys have the following special preset functions: (See also the section entitled "MS-DOS Editing and Function Keys" for a further explanation).

F1 The F1 key copies and displays the previously-entered line, one character at a time.

F2

The F2 key, when followed by a character, copies and displays the previously-entered line up to the first occurence of the specified character.

F3

The F3 key copies and displays the entire previously-entered line.

F4

The F4 key, when followed by a character, copies (but does not display) the first occurence of the specified character plus the remainder of the previously-entered line.

F5

The F5 key moves the cursor down one line and allows you to edit or retype the current line.

F6

The F6 key generates an end-of-file character (Ctrl-Z) for use with some applications.

Del

The Del key deletes the character at the cursor position in the previously-entered line. After pressing Del, press F3 to copy the remainder of the line. Note that if you press the Del key several times, the same amount of characters are deleted from the previously-entered line. Press F3 to display and enter the remainder of the line.

Ins

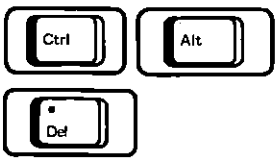
The Ins key toggles the display to the insert mode, which allows you to copy part of the previously-entered line, insert additional characters. When ENTER is pressed, the insert mode is toggled back to the non-insert mode.

Multiple key functions

CONTROL-BREAK (Ctrl, Break)

If the Ctrl key is held down and the Break key is pressed, the listing of a long program in BASIC or another application can be aborted.

SYSTEM RESET (Ctrl, Alt, Del)



When the Ctrl, Alt, and Del keys are all pressed simultaneously, the computer is reset and MS-DOS is booted as if the power was turned on.

ECHOING (Ctrl, PrtSc)

If the Ctrl key is held down and the PrtSc key is pressed, a running record of all commands input to the computer is output to the printer.

HARDCOPY (SHIFT, PrtSc)

If the SHIFT key is held down and the PrtSc key is pressed, a hard copy of the screen is printed on the printer.

STOP SYSTEM (Ctrl,NumLock)

If the Ctrl key is held down and the NumLock key is pressed,the computer stops all of its operation and waits for another key press.

Note:Once STOP SYSTEM function is executed,the redlamp of NumLock key may glow,although the numeric keypad retains its cursor control function. To recover this condition,execute STOP SYSTEM function once agein.

Running the operating system

To start your computer so that you can begin to run programs on it, you must first initialize your computer. This initialization process is called "cold start", and it loads the operating system into the computer.

Turn on your computer, insert a system disk in drive A (the drive to the left), and close the drive door. The disk goes into the drive with the write protect seal down. After a few seconds, the following prompt will appear:

```
MS-DOS Version x.xx
Copyright xxxx,xx Microsoft Corp.

Command V. x.xx
Current date is xxx xx-xx-xxxx
Enter new date:
```

This prompt appears whenever you turn on the computer or reset the computer. The computer is now waiting for you to input the current date. Type the date in the form mm-dd-yy where:

mm is a one- or two-digit number between 1 and 12 representing the month.

dd is a one- or two-digit number between 1 and 31 representing the day.

yy is a two- or four-digit number between 80 and 99 representing (omitting the 19) the years 1980 to 1999 or between 1980 and 2099.

The numbers in the date are seperated by hyphens (-) or slashes (/). For example, to input the date July 6, 1984, type either 07-06-84 or 07/06/84. After you input the date in this format, press the ENTER key to proceed to the next step.

Note that the computer will check for the possibility of an incorrect date being entered. For example, you cannot enter thirty-one for the day if the month specified has only thirty days.

Next a prompt will appear that shows the current computer time:

```
Current time is hh:mm:ss.cc
Enter new time:
```

Press the ENTER key immediately if you do not want to change the time. To change the time shown, type the time in the form hh:mm, where:

hh is a one- or two-digit number between 0 and 23 representing the hour on a twenty-four hour clock.

mm is a one- or two-digit number between 0 and 59 representing the minutes.

Note that you do not need to enter the seconds (ss) or the hundredths of seconds (cc).

The entries for the hours and minutes must be separated by colons (:). Press the ENTER key after you enter the time. The time that you enter here replaces the system time.

If you do not enter the time correctly, the computer will display the following message:

```
Invalid time
Enter new time:
```

Re-enter the time with the correct syntax or press the ENTER key to skip entering the system time.

The computer uses this date and time information to keep track of when the information in a file was last altered. Note that this computer clock is not always accurate because the clock stops whenever a floppy disk drive is being used.

The computer should now be at the command level where you will see the following prompt:

```
A>
```

Whenever you see this prompt, the computer is ready to accept commands. This is called the disk operating system (DOS) mode.

When in DOS mode, you can run your application software. To display the files on the disk, enter DIR and press ENTER. The DIR command displays the disk files.

File names are a maximum of eight characters, and an optional period plus a maximum of three more characters. The first eight characters are called the "file name". The three extra characters at the end is called the "file name extension". The files that have the extensions .COM and .EXE are command files that hold the application software and system commands. To run software having these extensions (.COM and .EXE), simply type the filename (without the extension) and press ENTER.

If you have a two-disk drive computer system, place your application software on the second disk drive (drive B:, located on the right). Input B: and press ENTER to change the primary drive to drive B. (To confirm the files on the disk, enter DIR command.) Most application software can be run on drive B just as you are running them on drive A.

If you have a single drive system, simply remove the system disk from the disk drive and replace it with the disk having the application software, then run them by specifying their filenames.

Formatting your disks

Before a new disk can be used with the computer, it must first be formatted. Formatting is a process that writes some information on the disk so that files can be stored on it. This information is used by the computer to help it keep track of the absolute locations of the data files (which are magnetically stored) on the disk.

MS-DOS has a command called FORMAT which will format your disks. The FORMAT command is described in more detail in Chapter 5. Refer to Chapter 5 or the MS-DOS reference manual which can be purchased under separate cover from your Sanyo dealer.

CAUTION: THE FORMAT COMMAND DESTROYS ANY INFORMATION PREVIOUSLY WRITTEN ON THE DISK. BE SURE TO USE NEW DISKS OR DISKS THAT HAVE DATA THAT YOU NO LONGER NEED AS THE ENTIRE CONTENTS OF THE DISK WILL BE ERASED DURING THIS PROCESS.

Formatting a disk with a double drive system
With the system disk in drive A, type in the following FORMAT command and press ENTER:

```
A>FORMAT B: /S
```

/S is an option of the FORMAT command that automatically copies the file containing the operating system onto the disk after it is formatted. The following files are copied when the /S option is added: IO.SYS, MSDOS.SYS, and COMMAND.COM. However, the first two files will not be displayed when the disk directory is listed.

When the FORMAT command is entered, the following prompt will appear:

```
Insert new disk for drive B:
and strike any key when ready
```

Place a new disk in drive B and press ENTER. The formatting process then begins, and is complete when the following prompt is displayed:

```
Format another (Y/N)?
```

If you want to format another disk, remove the formatted disk from the drive, place another new disk in the drive, input "Y" and press ENTER.

If you do not want to format another disk, input N and press ENTER. This returns you to the MS-DOS command level.

Formatting a disk with a single drive system
With the system disk in drive A, type in the following FORMAT command and press ENTER:

```
A>FORMAT /S
```

/S is an option of the FORMAT command that automatically copies the file containing the operating system onto the disk after it is formatted. The following files are copied when the /S option is added: IO.SYS, MSDOS.SYS, and COMMAND.COM. However, the first two files will not be displayed when the disk directory is listed.

When the FORMAT command is entered, the following prompt will appear:

```
Insert new disk for drive A:
and strike any key when ready
```

Remove the system disk from the drive. Place a new disk in drive A and press ENTER. The formatting process then begins, and is complete when the following prompt is displayed:

```
Format another (Y/N)?
```

If you want to format another disk, remove the formatted disk from the drive, place another new disk in the drive, input"Y"and press ENTER.

If you do not want to format another disk, input"N"and press ENTER. This returns you to the MS-DOS command level.

How to copy your files

COPY command

One reason why photocopy companies have become huge businesses is that people working in organizations need to make copies. When using a computer, you will often need more than one copy of a disk file. It is a good idea to make a backup copy of your programs in case something happens to your original. Software is valuable and the COPY command allows you to make copies of your files.

Disk files are handled by their filenames, in the COPY command, if you copy a file onto the same disk, you need to create a new file name for the backup. If you copy files onto a second disk, you do not have to specify a new filename.

The format for the COPY command is as follows:

COPY *drive* : *original filename* [*drive* :*backup filename*]

Examples:

COPY A:OLD.TXT A:NEW

A:*.* B: (Copies all)

You have made a copy of your file named OLD.TXT on drive A to the file named NEW on the same drive. Note that you do not always need to specify the three letter file extension.

COPY A:MY.TXT B:MY.TXT

The above command copies the file named MY.TXT on drive A to the drive B. Note that the file specification, since it includes two different drives, is still unique. When you use two different drives with the COPY command, you do not have to invent a new filename. As such, part of the above command is redundant, and you can shorten it to the following:

COPY A:MY.TXT B:

DISKCOPY command
The DISKCOPY command copies entire disks. This is faster than using the COPY command during the cases when you want to copy all the contents on one disk to another.

AS WITH USING THE FORMAT COMMAND, THE DISKCOPY COMMAND ERASES ALL THE INFORMATION ON THE TARGET DISK. BE SURE TO USE A NEW DISK OR A DISK WHICH HAS INFORMATION WHICH YOU NO LONGER NEED.

The target disk needs to be formatted before this command is used.

Making a disk copy with a double drive system
To make a copy of a disk using a double drive system, type:

```
DISKCOPY A: B:
```

The system will display the following prompt.

```
Insert source disk in drive A:
Insert formatted target disk into drive B:
Strike any key when ready
```

Place the disk to be copied in the drive A and a blank disk in drive B. Press ENTER. The computer will make a copy of the source disk onto the target disk. This process will take a few minutes.

When the disk copy is complete, the following prompt will appear:

```
Copy complete
Copy another (Y/N)?
```

Type Y if you want to copy other disks or make another copy of the source disk.

Note that the target disk must be formatted in the same format as the source disk. This must be done before using the DISKCOPY command. If the formats differ, the following error message appears.

```
Source and target disks are not the
same format. Cannot do the copy.
```

Also note that the DISKCOPY command cannot be used if the source disk has been damaged because the computer cannot read the damaged portions. In this case use the COPY command to make copies of the parts of the source disk that are undamaged.

Making a disk copy with a single drive system
To make a copy of a disk using a single drive system, type:

```
DISKCOPY
```

The system will display the following prompt.

```
Insert source disk in drive A:
Strike any key when ready
```

Place the disk to be copied in the drive and press ENTER. The computer will read the part of the contents of the disk into its memory. When the following prompt appears,

```
Insert formatted target disk in drive A:
Strike any key when ready
```

Take the source disk out of the drive, and insert a new disk in the drive.

The above steps will be done repeated several times and when the copy is complete, the following prompt will appear:

```
Copy complete
```

Resetting the system

Sometimes an error in operation or in a program causes the keyboard to become inoperative. When this occurs, commands can no longer be input into the system. You can reset the computer in two ways:

1. Turn the power off and then back on after approximately 10 seconds.

When the power is applied, the system will start (cold boot). Note that any data that was not saved on the disk is lost when the power is turned off.

2. Reset the system with a warm start.

A warm start (reset) may be done by pressing the Ctrl, Alt, and Del keys simultaneously. Note that any data that was not saved on the disk is lost when the system is reset.

Turning the system power off

1. Remove the disks from the drives.

2. Turn the power off.

3. Store the AC connector cord in the compartment on the back of the main unit.

4. Attach the keyboard and lock it.

   Do this by opening the feet of the keyboard, clipping the keyboard cord onto the clips on the right side of the main unit, and placing the keyboard against the main unit. Lock the keyboard in place by closing its feet.

CHAPTER .2
SANYO GW-BASIC

Introduction 2-2
Overview 2-2
Syntax notation 2-2
Using the BASIC Interpreter 2-4
Starting BASIC 2-4
Modes of operation 2-4
Line format 2-5
Learning the Language 2-6
Character set 2-6
Special characters 2-6
Constants 2-7
String and numeric constants 2-7
Variables 2-8
Variable names and declaration characters 2-8
Array variables 2-9
Expressions and operators 2-10
Precedence of operations 2-10
Arithmetic operators 2-11
Integer division and modulus arithmetic 2-12
Overflow and division by zero 2-12
Relational operators 2-13
Logical operators 2-14
String operators 2-16
Type conversion 2-17
Functions 2-18
Intrinsic functions 2-18
User-defined functions 2-18
Writing Programs Using the Screen Editor 2-19
Full screen editor 2-19
Writing programs 2-19
Editing programs 2-20
Control characters 2-21
Editing lines with syntax errors 2-22
Working with Files 2-23
Default drive 2-23
Filenames and paths 2-23
Filename specifications 2-23
Pathnames 2-23
Working with pathnames in BASIC 2-25
Handling files 2-26
Program file commands 2-26

Introduction

Overview

BASIC is a general-purpose programming language that can be effectively used in many applications, including business, science, games, and education. BASIC has two modes, direct and indirect, that can be used interchangeably. In the indirect mode, programs can be entered for later execution. In the direct mode, commands can be entered for immediate execution. Because BASIC has two modes, a user can do calculations and test programs effectively.

Syntax notation

To simplify command syntax explanations, the following notation will be used in this manual.

[ ] Square brackets indicate optional entries that may be omitted.

Italic Italics indicate user entries. Type in an entry appropriate to the text; for example, type in a file name in *filename.*

{ } Braces indicate a choice between two or more entries. At least one of the enclosed entries must be chosen, unless the entries are also enclosed in square brackets, which would indicate that the entries are optional.

| Vertical bars are used to separate choices within the above braces.

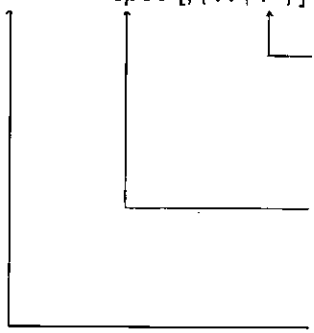
... Ellipses indicate that an entry may be repeated as many times as needed.

CAPS Capital letters indicate the portions of statements or commands that must be entered exactly as shown.

All other punctuation, such as commas, colons, slashes, and equals signs, must be entered exactly as shown.

Examples

SAVE *filespec* [, { A | P }]



These two entries are optional because they are enclosed in square brackets. They must also be typed as shown. The braces indicate a choice.

The italic *filespec* means that you must type the file specification (optional disk drive, filename and optional extension).

Capital letters mean that the command must be entered exactly as shown.

In the formats of the command, statement, function, or variable given in next chapter, some of the parameters are abbreviated as follows:

x, y, z	represent numeric expressions
i, j, k, m, n	represent integer expressions
x$, y$	represent string expressions
v, v$	represent numeric and string variables, respectivery

Using the BASIC Interpreter

Starting BASIC

BASICA is the name of the Sanyo BASIC interpreter. To start BASIC, enter the following command when in MS-DOS mode and A> is on the screen.

BASICA

To run a specific program immediately after BASIC has started, enter the following command:

BASICA *filespec*

where *filespec* is an optional device name, a filename, and an optional extension name.

For example, to run the program FILE.BAS on disk drive A:, enter:

BASICA A:FILE.BAS

Modes of operation

The BASIC Interpreter may be used in either of two modes: direct mode or indirect mode.

In direct mode, statements and commands are executed as they are entered. Statements in this mode are not preceded by line numbers. After each statement is executed by pressing the ENTER key, the screen displays the "Ok" prompt. The results of arithmetic and logical operations are displayed immediately and are stored for later use, but the instructions themselves are lost after execution. Direct mode is thus useful for debugging and for using the BASIC Interpreter for quick computations that do not require a complete program.

Indirect mode is used to enter programs. Program lines preceded by line numbers are stored in memory. The program in memory is then executed by entering the RUN command.

Line format

BASIC program lines have the following format (square brackets indicate optional input):

nnnnn BASIC statement [: BASIC *statement* ...] < ENTER *key* >

More than one BASIC statement may be placed on a line, and they must be separated by colons(:).

In the indirect mode, a BASIC program line always begins with a line number and ends with an ENTER key. Line numbers indicate the order in which the program lines are stored in memory. Line numbers are also used as references for branching and editing. Line numbers must be between 0 and 65529.

A line may contain a maximum of 255 characters.

A period (.) may be specified in the EDIT, LIST, AUTO, and DELETE commands to refer to the current line.

Learning the Language

As with any language, BASIC has an alphabet and common phrases. This section presents the BASIC character set and the rules for its constants, variables, and expressions.

Character Set

The BASIC character set consists of alphabetic, numeric, and special characters.

The alphabetic characters in BASIC are the upper- and lowercase letters of the English alphabet.
The numeric characters in BASIC are the digits 0 through 9. The alphabetic characters A, B, C, D, E, and F may be used in hexadecimal numbers.

Special Characters
The following special characters are recognized by BASIC:

Character	Action
	Blank
=	Equals sign or assignment symbol
+	Plus sign
-	Minus sign
*	Asterisk or multiplication symbol
/	Slash or division symbol
^	Up arrow or exponentiation symbol
(	Left parenthesis
)	Right parenthesis
%	Percent
#	Number (or pound) sign
$	Dollar sign
!	Exclamation point
[	Left bracket
]	Right bracket
,	Comma
.	Period or decimal point
'	Single quotation mark (apostrophe)
;	Semicolon
:	Colon
&	Ampersand

?	Question mark
<	Less than
>	Greater than
\	Backslash or integer division symbol
@	At sign
_	Underscore

Constants

Constants are values that will not be changed during execution. There are two types of constants: string and numeric.

String and numeric constants
String constants are sequences of up to 255 alphanumeric or control characters enclosed in double quotation marks.

Examples:

"HELLO"
"$25,000.00"
"Number of Employees"

Numeric constants are positive or negative numbers. There are five types of numeric constants:

1. Integer — Whole numbers between –32768 and 32767. Integer constants do not contain decimal points.

2. Fixed-point — Positive or negative real numbers; numbers that contain decimal points.

3. Floating-point — Positive or negative numbers represented in exponential form (similar to scientific notation). There are two types of floating-point constants; single-precision and double-precision.

Examples:
235.988E–7 = .0000235988
2359E6 = 2359000000
2.35988D–15
= .000000000000235988
235888D11
= 23598800000000000
(Double precision floating-point constants are written with the letter D instead of E.)

4. Hexadecimal — Hexadecimal numbers, denoted by the prefix &H. Hex constants may be no greater than &HFFFF (decimal 65535).

   Examples: &H76
   &H32F

5. Octal — Octal numbers, denoted by the prefix &O or &. Octal constants may not exceed &O177777 (decimal 65535).

   Examples: &O347
   &1234

Variables

Variables are names used to represent values used in a BASIC program. The value of a variable may be assigned by the programmer, or may be assigned as the result of calculations in the program. Before a variable is assigned a value, its value is assumed to be zero (or null for a string variable).

Variable names and declaration characters
BASIC variable names are character strings containing up to 40 characters. Variable names can contain letters, numbers, and the decimal point. The first character must be a letter. Special type declaration characters (%, !, #) are also allowed.

A variable name may not be a reserved word; however, reserved words that are embedded within variable names are allowed, with one exception: no variable may start with the letters USR. For example, a variable named USRNAM$ will generate a syntax error. Reserved words (See Chapter 6, BASIC Reserved Words)

include all BASIC commands, statements, function names, and operator names. If a variable begins with the letters FN, it is assumed to be a call to a user-defined function.

Variables may represent numeric values or strings. Names for string variables are written with a dollar sign ($) as the last character, for example: A$ = "SALES REPORT". The dollar sign is a variable type declaration character; that is, it "declares" that the variable will represent a string.

Numeric variable names may also declare the variable type to be integer, single precision, or double precision variables. The variable type declaration characters for these variable names are as follows:

%	Integer variable
!	Single precision variable
#	Double precision variable

If the variable type declaration character is not included, the default is single precision. However, if a number specified in a program has too many significant digits to be represented by a single precision variable, it will be represented as a double precision varable, and the "#" mark which signifies double precision will follow the number in the program listing.

Examples of BASIC variable names:

PI#	Double precision
MINIMUM!	Single precision
LIMIT%	Integer
N$	String
ABC	Single precision

The default variable type may be changed by using the BASIC statements DEFINT, DEFSTR, DEFDBL, and DEFSNG.

Array variables

An array is a group of values with the same variable name. Each element in an array is assigned a unique name with an integer or expression to represent an integer as a subscript. An array variable has as many subscripts as dimensions in the array. For

example, V(10) corresponds to a value in a one-dimension array, T(1,4) corresponds to a value in a two-dimension array, and so on.

Expressions and operators

Expressions may be string or numeric constants, variables, or a combination of constants and variables with operators. Expressions are always evaluated to produce single values.

Operators perform mathematical or logical operations on values. Operators in BASIC fall into three categories:

1. Arithmetic
2. Relational
3. Logical

The categories are described in the following sections.

Precedence of operations

Operators in BASIC have an order of precedence; that is, when there are several operations within the same program statement, certain types of operations will be evaluated before others. If the operations are equal in precedence, the leftmost operation will be executed first and the operations to the right will be executed in order so that the rightmost operation is executed last. The following is the order in which operations are executed.

1. Exponentiation
2. Negation
3. Multiplication and Division
4. Integer Division
5. Modulus Arithmetic
6. Addition and Subtraction
7. Relational Operators
8. NOT
9. AND
10. OR & XOR
11. EQV
12. IMP

Arithmetic operators
The arithmetic operators, in order of precedence, are:

Operator	Operation	Sample expression
^	Exponentiation	X^Y
–	Negation	–X
*, /	Multiplication and floating-point division	X*Y X/Y
\	Integer division	12\6=2
MOD	Modulus arithmetic	10 MOD 4=2 (10/4=2 with remainder 2)
+, –	Addition and subtraction	X+Y

You can clarify or change the order of evaluation by using parentheses. Operations within parentheses are performed first. Within parentheses, the above order of operations is maintained.

The following list gives examples of algebraic expressions and how to express them in GW-BASIC.

Algebraic expression	BASIC expression	
X+2Y	X+Y*2	
$X-\frac{Y}{Z}$	X–Y/Z	
$\frac{XY}{Z}$	X*Y/Z	
$\frac{X+Y}{Z}$	(X+Y)/Z	
X^Y	X^Y	
X(–Y)	X*(–Y)	Two consecutive operators must be separated by parentheses.

Integer division and modulus arithmetic
In addition to the above standard operators, GW-BASIC supports integer division and modulus arithmetic.

Integer division is specified by using the backslash (\). The operands are rounded to integers (which must be between –32768 to 32767) before the division is done, and the quotient is truncated to an integer.

For example,

```
100 LET DIV1 = 10\4
200 LET DIV2 = 25.68\6.99
300 PRINT DIV1, DIV2
```

will yield

```
 2     and      3.
```

Modulus arithmetic is specified by using the operator MOD. Modulus arithmetic yields the integer value that is the remainder of integer division.

For example,

```
10 PRINT 10.4 MOD 4, 25.68 MOD 6.99
```

will yield

```
2     and      5.
```

because (10/4=2, with a remainder of 2) and (26/7=3, with a remainder of 5).

Overflow and division by zero
If division by zero is attempted during evaluation of an expression, a "Division by zero" error message is displayed. Infinity (defined here as the largest number that can be represented on the computer in floating-point format) with the sign of the numerator will be supplied as the result of the division. If the evaluation of an exponential operator results in zero being raised to a negative power, the "Division by zero" error message is

displayed and positive machine infinity is supplied as the result of the exponentiation.

If an overflow occurs, the interpreter displays an "Overflow" error message and supplies infinity with the algebraically correct sign as the result.

Relational operators

Relational operators are used to compare two values. The result of the comparison is either "true" (–1) or "false" (0). This result may then be used to make decisions on program flow. (See the IF Statement.)

The relational operators are:

Operator	Relationship	Example
=	Equality	X=Y
<>	Inequality	X<>Y
<	Less than	X<Y
>	Greater than	X>Y
<=	Less than or equal to	X<=Y
>=	Greater than or equal to	X>=Y

(The equals sign is also used to assign a value to a variable.)

When arithmetic and relational operators are combined in one expression, the arithmetic is always done first. For example, the expression

```
X+Y<(T-1)/Z
```

is true if the value of X plus Y is less than the value of T–1 divided by Z.

More examples:

```
IF SIN(X)<0 GOTO 1000
IF I MOD J<>0 THEN K=K+1
```

Logical operators

The logical operator performs bit-by-bit calculation and returns a result which is either "true" (not zero) or "false" (zero). In an expression, logical operations are performed after arithmetic and relational operations. The operators are listed in order of precedence.

Operation	Value	Value	Result
NOT			
	X		**NOT X**
	T		F
	F		T
AND			
	X	**Y**	**X AND Y**
	T	T	T
	T	F	F
	F	T	F
	F	F	F
OR			
	X	**Y**	**X OR Y**
	T	T	T
	T	F	T
	F	T	T
	F	F	F
XOR			
	X	**Y**	**X XOR Y**
	T	T	F
	T	F	T
	F	T	T
	F	F	F
EQV			
	X	**Y**	**X EQV Y**
	T	T	T
	T	F	F
	F	T	F
	F	F	T

IMP

X	Y	X IMP Y
T	T	T
T	F	F
F	T	T
F	F	T

Just as relational operators can be used for decisions regarding program flow, logical operators can connect two or more relations and return a true or false value to be used in a decision (see IF Statements).

Example:

```
IF D<200 AND F<4 THEN 80
IF I>10 OR K<0 THEN 50
IF NOT P THEN 100
```

Logical operators work by converting their operands to 16-bit, signed, two's complement integers in the range -32768 to 32767. (If the operands are not in this range, an error results.) If both operands are supplied as 0 or –1, logical operators return 0 or –1. The given operation is performed on these integers bit-by-bit; i.e., each bit of the result is determined by the corresponding bits in the two operands.

The following examples, all using decimal numbers, demonstrate how the logical operators work.

63AND16=16	63 = binary 111111 and 16 = binary 10000, so 63 AND 16 = 16.
–1 AND 8 = 8	–1 = binary 1111111111111111 and 8 = binary 1000, so –1 AND 8 = 8.
4 OR 2 = 6	4 = binary 100 and 2 = binary 10, so 4 OR 2 = 6 (binary 110).
–1 OR –2 = –1	–1 = binary 1111111111111111 and –2 = binary 1111111111111110, so –1 OR –2 = –1. The bit complement of sixteen zeros is sixteen ones, which is the two's complement of –1.

String operators
Strings may be concatenated by using the plus sign (+). For example:

```
10 A$="FILE" : B$="NAME"
20 PRINT A$+B$
30 PRINT "NEW "+A$+B$
RUN
FILENAME
NEW FILENAME
Ok
```

Strings may be compared by using the same relational operators that are used with numbers:

```
=    <>    <    >   <=    >=
```

String are compared by taking one character at a time from each string and comparing the ASCII codes. If all the ASCII codes are the same, the strings are equal. If the ASCII codes differ, the lower code number precedes the higher. If during string comparison, the end of one string is reached, the shorter string is said to be smaller. The leading and trailing blanks are significant.

For example:

"FILENAME"	is equal to	"FILENAME"
"X&"	is greater than (because # comes before &)	"X#"
"CL "	is greater than (because of the trailing space)	"CL"
"SMYTH"	is less than	"SMYTHE"
B$	is less than (where B$="8/12/78")	"9/12/78"

String comparisons can thus be used to test string values or to alphabetize strings.

Type conversion

When necessary, BASIC will convert a numeric constant from one type to another. The following rules and examples apply to conversions.

1. If a numeric variable of one type is specified to a numeric constant of a different type, the number will be stored as the type declared in the variable name.

   Example:

   ```
   10 PERCENT%=23.42
   20 PRINT PERCENT%
   RUN
     23
   Ok
   ```

2. During evaluation of an expression, all of the operands in an arithmetic or relational operation are converted to the same degree of precision as that of the most precise operand. Also, the result of an arithmetic operation is returned to this degree of precision.

3. Logical operators convert their operands to integers and return an integer result. Operands must be between –32768 and 32767, or an "Overflow" error occurs.

4. When a floating-point value is converted to an integer, the fractional portion is rounded.

   Example:

   ```
   10 CASH% = 55.88
   20 PRINT CASH%
   RUN
     56
   Ok
   ```

Functions

BASIC has functions included within it (intrinsic functions).
The user can also define functions.

Intrinsic functions
When a function is used in an expression, a predetermined operation is done on an operand. BASIC has functional operators within the system, such as SQR (square root) or SIN (sine), and these resident functions are called "Intrinsic functions".

User-defined functions
BASIC also allows "user-defined" functions, these functions are written into the program by the programmer. See the DEF FN Statement.

Writing Programs Using the Screen Editor

There are two ways to enter and edit BASIC text: you can issue an EDIT command to enter edit mode or use the full screen editor.

Full screen editor

The full screen editor gives you immediate visual feedback, so that the program text is entered in a "what you see is what you get" manner. If the user has a program listing on the screen, the cursor can be moved to a program line, the line edited, and the change entered by pressing the ENTER key.

With the full screen editor, you can move quickly around the screen, making corrections where necessary. The changes are entered by placing the cursor on the changed line and pressing ENTER key.

Writing programs
The full screen editor can be used any time after the BASIC interpreter displays the "Ok" prompt. Any line of text that is then entered is processed by the editor. In BASIC, a line must be preceded by a number.

Program statements are processed by the editor in one of the following ways:

1. A new line is added to the program. This occurs if the line number is a valid program line number (a number between 0 through 65529), the number does not exist in the program already, and at least one non-blank character follows the line number.

2. An existing line is modified. This occurs if the line number matches that of an existing line in the program. The existing line is replaced with the new line.

3. An existing line is deleted. This occurs if the line contains only the line number, and the number matches that of an existing line.

4. The statements are passed to the BASIC interpreter (i.e., the statement is executed).

5. An error is produced.
   If an attempt is made to delete a non-existent line, an "Undefined line number" error message is displayed.
   If program memory is exhausted, and a line is added to the program, an "Out of memory" error message is displayed, and the line is not added.

Some BASIC keywords can be input by fewer key strokes. For example, the word "PRINT" can be input by ALT–P (hold the ALT key down, type P, and release the ALT key).

ALT–A	AUTO	ALT–N	NEXT
ALT–B	BSAVE	ALT–O	OPEN
ALT–C	COLOR	ALT–P	PRINT
ALT–D	DELETE	ALT–Q	(not assigned)
ALT–E	ELSE	ALT–R	RUN
ALT–F	FOR	ALT–S	SCREEN
ALT–G	GOTO	ALT–T	THEN
ALT–H	HEX$	ALT–U	USING
ALT–I	INPUT	ALT–V	VAL
ALT–J	(not assigned)	ALT–W	WIDTH
ALT–K	KEY	ALT–X	XOR
ALT–L	LOCATE	ALT–Y	(not assigned)
ALT–M	MID$	ALT–Z	(not assigned)

Editing programs

Use the LIST command to display an entire program or a range of lines on the screen for modification with the full screen editor. Text can then be modified by moving the cursor to where the change is needed and then doing one of the following:

1. Typing over existing characters
2. Deleting characters to the right of the cursor
3. Deleting characters to the left of the cursor
4. Inserting characters
5. Appending characters to the end of the line

These actions are performed by special keys assigned to the various full screen editor functions (see the next section).

Changes to a line are recorded when a ENTER key is pressed while the cursor is somewhere on that line. The ENTER key stores all changes for that line, up to the 255 character line limitation, no matter where the cursor is located on the line.

Control characters

Next table lists the BASIC control characters and summarizes their functions. The Control-key sequence normally assigned to each function is also listed.

Control

Key	Equivalent	Function
CTRL–B	Ctrl– ←	Move cursor to start of previous word
CTRL–E	Ctrl–End	Truncate line (clear text to end of line)
CTRL–F	Ctrl →	Move cursor to start of next word
CTRL–G		Beep
CTRL–H	BS (⇦)	Backspace, deleting characters passed over
CTRL–I	TAB (⭾)	Tab (8 spaces)
CTRL–J		Line feed
CTRL–K	Home	Move cursor to home position at top left hand corner of screen
CTRL–L	Ctrl–Home	Clear screen
CTRL–M	ENTER (↵)	Enter current logical line
CTRL–N	End	Append to end of line
CTRL–R	Ins	Toggle betwen insert and typeover modes
CTRL–S		Suspend or restart program
CTRL–T		Display function key contents
CTRL–U	Esc	Clear logical line
CTRL–W		Delete word
CTRL–X		Display previous program line
CTRL–Y		Display following program line
CTRL–Z		Clear to end of window
CTRL–\	→	Cursor right
CTRL–]	←	Cursor left
CTRL–^	↑	Cursor up
CTRL–_	↓	Cursor down (underscore)

Editing lines with syntax errors

When a syntax error is encountered during program execution, BASIC prints the line containing the error and enters direct mode. You can correct the error, enter the change, and reexecute the program. Thus, if you wish to examine the values of variables at the time the syntax error was encountered, examine the values before modifying the program line.

Working with Files

This section discusses how files are used and called in BASIC.

Default drive

When a file is specified (in commands or statements such as FILES, OPEN, KILL), the default (current) disk drive is considered to be the one that was the default in MS-DOS before BASIC was called.

Filenames and paths

BASIC uses MS-DOS version 2's enhanced directory structure, allowing files to be accessed through their pathnames (See below).

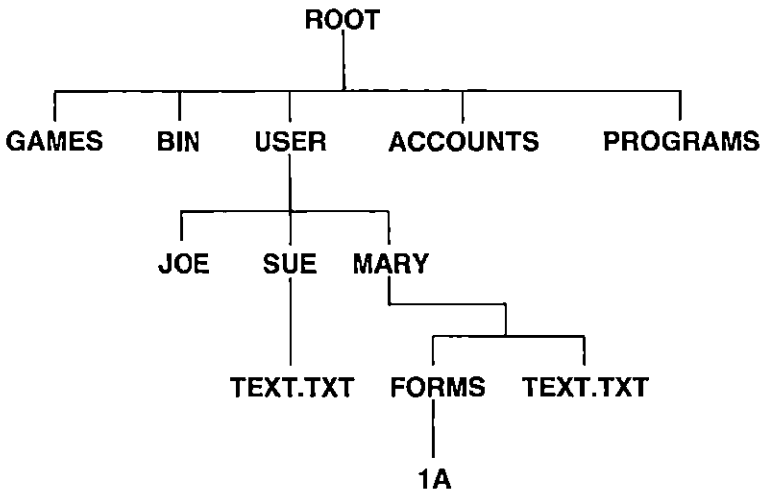
Filename specifications
Files are specified using the naming conventions of MS-DOS version 2.XX. All filespecs may begin with a device specification such as A: or B: or COM1: or LPT1:. If no device is specified, the current drive is assumed. The default extension .BAS is appended to filenames in LOAD, SAVE, MERGE and RUN *filename* commands, providing that no period (.) appears in the filespec and if the filename is less than nine characters long.

Examples:

```
RUN "NEWFILE.BAS"
RUN "A:NWFILE.BAS"
SAVE "NEWFILE" (file is saved with .BAS extension on
default drive)
```

Pathnames
A pathname is a sequence of directory names followed by a simple filename, each separated from the previous one by a backslash (\). The pathname must be no longer than 128 characters. If a device is specified, it must be specified at the beginning of the pathname. A simple filename is a sequence of characters optionally preceded by a drive designation, optionally followed by an extension, and containing no backslashes.

[*d* :][\][*directory* \][*directory...*\][*filename*]



A sample hierarchical directory structure

The two entries named TEXT.TXT, and the entry named 1A are files.

If a pathname begins with a backslash, MS-DOS searches for the file beginning at the root (or top) of the tree. Otherwise, MS-DOS begins at the user's current directory, (the working directory), and searches downward from there.

The pathname of SUE's TEXT.TXT file is \USER\SUE\TEXT.TXT.

When you are in your working directory, a filename and its corresponding pathname may be used interchangeably. Some sample names are:

\ Indicates the root directory.

\PROGRAMS
Sample directory under the root directory containing program files.

\USER\MARY\FORMS\1A
A typical full pathname. This one happens to be a file named 1A in the directory named FORMS belonging to the a subdirectory of USER named MARY.

USER\SUE
A relative pathname; it names the file or directory SUE in subdirectory USER of the working directory. If the working directory is the root, it names \USER\SUE.

TEXT.TXT
Name of a file or directory in the working directory.

MS-DOS provides special shorthand notations for the working directory and the parent directory (one level up) of the working directory:

. MS-DOS uses this shorthand notation to indicate the name of the working directory in all hierarchical directory listings. MS-DOS automatically creates this entry when a directory is made.

.. The shorthand name of the working directory's parent directory. If you type:

```
DIR ..
```

then MS-DOS will list the files in the parent directory of your working directory.

Working with pathnames in BASIC
Not only does BASIC provide the ability to access files from other directories using pathname approaches, but it can also be used to create, change, and remove paths, using the BASIC commands MKDIR, CHDIR, and RMDIR.

The BASIC statement MKDIR "ACCOUNTS" would create a new directory, ACCOUNTS, in the working directory of the current drive.

The BASIC statement CHDIR "B:EXPENSES" would change the current directory on B: to EXPENSES.

The BASIC statement RMDIR "CLIENTS" would delete an existing directory, CLIENTS, as long as that directory did not have any files, with the exception of "." and "..".

Handling files

File I/O procedures for the beginning BASIC user are examined in this section. If you are new to BASIC, or if you are encountering file-related errors, read through these procedures and program examples to make sure you are using all the file statements correctly.

Program file commands

The following is a review of the commands and statements used in program file manipulation. All file specifications may include the device and pathname.

SAVE *filespec* — Writes the program that currently resides in memory to the specified file.

LOAD *filespec* — Loads the program from file into memory. LOAD always deletes the current contents of memory and closes all files before loading.

RUN *filespec* — Loads the program from file into memory and runs it. RUN deletes the current contents of memory and closes all files before loading the program.

MERGE *filespec* — Loads the program file into memory but does not delete the current contents of memory. The program line numbers in the file are merged with the line numbers in memory. If two lines have the same number, the line from the file being merged is used.

After a MERGE command is executed, the "merged" program resides in memory, and BASIC returns to command level. In order to successfully MERGE a program, the *filespec* must have been saved in ASCII format.

KILL *filespec* — Deletes the file from the disk. *filespec* can be a program file or a data file.

NAME *old filespec* AS *new filespec*
Changes the name of a file. NAME can be used with a program file or a data file.

CHAPTER 3.
BASIC COMMANDS, FUNCTIONS, AND STATEMENTS

ABS Function 3-5
ASC Function 3-5
ATN Function 3-5
AUTO Command 3-5
BEEP Statement 3-6
BLOAD Command 3-6
BSAVE Command 3-6
CALL Statement 3-7
CDBL Function 3-7
CHAIN Statement 3-7
CHDIR Statement 3-8
CHR$ Function 3-8
CINT Function 3-8
CIRCLE Statement 3-8
CLEAR Statement 3-9
CLOSE Statement 3-9
CLS Statement 3-9
COLOR Statement 3-10
COM(*n*) Statement 3-11
COMMON Statement 3-11
CONT Command 3-11
COS Function 3-11
CSNG Function 3-11
CSRLIN Function 3-12
CVI, CVS, CVD Functions 3-12
DATA Statement 3-12
DATE$ Statement 3-12
DATE$ Variable 3-13
DEF FN Statement 3-13
DEFINT/SNG/DBL/STR Statements 3-13
DEF SEG Statement 3-13
DEF USR Statement 3-14
DELETE Command 3-14
DIM Statement 3-14
DRAW Statement 3-15
EDIT Command 3-16
END Statement 3-16
ENVIRON Statement 3-16
ENVIRON$ Function 3-16
EOF Function 3-17

ERASE Statement 3-17
ERDEV, ERDEV$ Functions 3-17
ERR and ERL Variable 3-17
ERROR Statement 3-17
EXP Function 3-18
FIELD Statement 3-18
FILES Statement 3-18
FIX Function 3-18
FOR...NEXT Statement 3-19
FRE Function 3-19
GET Statement - Files 3-19
GET Statement - Graphics 3-19
GOSUB...RETURN Statement 3-20
GOTO Statement 3-20
HEX$ Function 3-20
IF...THEN [...ELSE] / IF...GOTO Statements 3-20
INKEY$ Function 3-21
INP Function 3-21
INPUT Statement 3-21
INPUT# Statement 3-21
INPUT$ Function 3-21
INSTR Function 3-22
INT Function 3-22
KEY Statement 3-22
KEY(*n*) Statement 3-23
KILL Command 3-23
LEFT$ Function 3-23
LEN Function 3-23
LET Statement 3-24
LINE Statement 3-24
LINE INPUT Statement 3-24
LINE INPUT# Statement 3-24
LIST Command 3-25
LLIST Command 3-25
LOAD Command 3-25
LOC Function 3-25
LOCATE Statement 3-26
LOF Function 3-26
LOG Function 3-26
LPOS Function 3-27
LPRINT and LPRINT USING Statements 3-27
LSET and RSET Statements 3-27
MERGE Command 3-28
MID$ Statement 3-28
MID$ Function 3-28

MKDIR Statement 3-28
MKI$, MKS$, MKD$ Functions 3-29
NAME Command 3-29
NEW Command 3-29
OCT$ Function 3-29
ON COM(*n*) Statement 3-30
ON ERROR GOTO Statement 3-30
ON...GOSUB and ON...GOTO Statements 3-30
ON KEY(*n*) Statement 3-31
ON PEN Statement 3-31
ON PLAY(*n*) Statement 3-31
ON STRIG(*n*) Statement 3-32
ON TIMER(*n*) Statement 3-32
OPEN Statement 3-33
OPEN COM Statement 3-34
OPTION BASE Statement 3-35
OUT Statement 3-35
PAINT Statement 3-35
PEEK Function 3-36
PEN ON, PEN OFF, PEN STOP Statements 3-36
PEN Function 3-36
PLAY Statement 3-37
PLAY Function 3-38
PLAY ON, PLAY OFF, PLAY STOP Statements 3-38
PMAP Function 3-38
POINT Function 3-39
POKE Statement 3-39
POS Function 3-39
PRESET Statement 3-40
PRINT Statement 3-40
PRINT USING Statement 3-41
PRINT# and PRINT# USING Statements 3-42
PSET Statement 3-43
PUT Statement - Files 3-43
PUT Statement - Graphics 3-44
RANDOMIZE Statement 3-44
READ Statement 3-44
REM Statement 3-44
RENUM Command 3-45
RESET Command 3-45
RESTORE Statement 3-45
RESUME Statement 3-46
RETURN Statement 3-46
RIGHT$ Function 3-46
RMDIR Statement 3-46

RND Function 3-47
RUN Statement/Command 3-47
SAVE Command 3-47
SCREEN Statement 3-48
SCREEN Function 3-48
SGN Function 3-49
SHELL Statement 3-49
SIN Function 3-49
SOUND Statement 3-50
SPACE$ Function 3-50
SPC Function 3-50
SQR Function 3-50
STICK Function 3-51
STOP Statement 3-51
STR$ Function 3-51
STRIG(*n*) Statements 3-52
STRIG(*n*) Function 3-52
STRING$ Function 3-53
SWAP Statement 3-53
SYSTEM Command 3-53
TAB Function 3-53
TAN Function 3-53
TIME$ Statement 3-54
TIME$ Variable 3-54
TIMER Statements 3-54
TRON, TROFF Commands 3-55
USR Function 3-55
VAL Function 3-55
VARPTR Function 3-56
VARPTR$ Function 3-56
VIEW Statement 3-57
VIEW PRINT Statement 3-57
WAIT Statement 3-57
WHILE...WEND Statements 3-58
WIDTH Statement 3-58
WINDOW Statement 3-59
WRITE Statement 3-59
WRITE# Statement 3-59

ABS Function

Format: *v* = ABS(*x*)

Purpose: To return the absolute value of the expression *x*.

ASC Function

Format: *v* = ASC(*x$*)

Purpose: To return a numerical value that is the ASCII code for the first character of the string *x$*.
(See Chapter 6 for ASCII codes.)

ATN Function

Format: *v* = ATN(*x*)

Purpose: To return the arctangent of *x* ,where *x* is in radians. Result is in the range –pi/2 to pi/2 radians.

AUTO Command

Format: AUTO [*line number*][,[*increment*]]

where *line number* is the number which will be used to begin numbering lines.
increment is the value that will be added to each line number to get the next line number.

Purpose: To automatically generate line numbers each time you press the ENTER key.

BEEP Statement

Format: BEEP

Purpose: To sound the speaker.

BLOAD Command

Format: BLOAD *filespec* [, *offset*]

offset is the offset address at which loading is to start in the segment.

Purpose: To load a specified memory image file into memory from any input device.

BSAVE Command

Format: BSAVE *filespec* , *offset* , *length*

offset is the offset address to start saving from in the segment.
length is a numeric expression in the range 1 to 65535. This is the length in bytes of the memory image to be saved.

Purpose: To transfer the contents of the specified area of memory to any output device.

CALL Statement

Format: CALL *numeric variable* [(*variable* [, *variable*] ...)]

where *numeric variable* contains an address that is the starting point in memory of the subroutine.
variable is the name of a variable contains the arguments that are passed to the external subroutine.

Purpose: To call a machine (or assembly) language subroutine.

CDBL Function

Format: *v* = CDBL(*x*)

Purpose: To convert *x* to a double precision number.

CHAIN Statement

Format: CHAIN [MERGE] *filespec* [,[*line number exp*] [, ALL][, DELETE *range*]]

line number exp is the starting point for execution of the called program.

Purpose: To call a program and pass variables to it from the current program.

CHDIR Statement

Format: CHDIR *pathname*

Purpose: To change the current operating directory.

CHR$ Function

Format: *v$* = CHR$(*n*)

Purpose: To convert an ASCII code to its character equivalent.

CINT Function

Format: *v* = CINT(*x*)

Purpose: To convert *x* to an integer by rounding the fractional portion.

CIRCLE Statement

Format: CIRCLE [STEP](*xcenter* , *ycenter*), *radius* [, *color* [, *start* , *end* [, *aspect*]]]

Purpose: To draw an ellipse or circle with the specified center and radius.

CLEAR Statement

Format: CLEAR [,[*Highest memory*][, *stack space*]]

Purpose: To set all numeric variables to zero, all string variables to null, and to close all open files; and, optionally, to set the end of memory and the amount of stack space.

CLOSE Statement

Format: CLOSE [[#] *file number* [,[#] *file number ...*]]

Purpose: To conclude I/O to a file.

CLS Statement

Format: CLS [*n*]

where *n* is in the range 0 to 2.

Purpose: To clear the screen.

n = 0 Clears both graphics and text modes
n = 1 Clears area in the viewport specified by VIEW statement
n = 2 Clears area in the text window made in text mode

COLOR Statement

Format: Text Mode

COLOR [*foreground*][,[*background*][, *border*]]

Medium Resolution Graphics Mode

COLOR [*background*][,[palette]]

Purpose: To select the foreground, background, and border colors.

In Text mode, *foreground* and *border* can be chosen from the following color codes:

0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Brown	14	Yellow
7	White	15	Hight intensity White

If a number equal to 16 plus the above color codes is used for *foreground* , the character will blink.

background can be any value between 0 and 7 above.

In Medium Resolution Graphics Mode, the value in *background* can be any of the above color codes.

For *palette* specify the following color code used with PSET, PRESET, LINE, CIRCLE, PAINT, and DRAW.

color	palette 0	palette 1
1	Green	Cyan
2	Red	Magenta
3	Brown	White

COM(*n*) Statement

Format: COM(*n*) ON
COM(*n*) OFF
COM(*n*) STOP

Where *n* is the number of the communications port. The range for *n* depends on the number of ports installed.

Purpose: To enable or disable event trapping of communications on the specified port.

COMMON Statement

Format: COMMON *variable* [, *variable*] ...

Purpose: To pass variables to a chained program.

CONT Command

Format: CONT

Purpose: To resume program execution after a break.

COS Function

Format: *v* = COS(*x*)

Purpose: To return the cosine of *x* , where *x* is in radians.

CSNG Function

Format: *v* = CSNG(*x*)

Purpose: To convert *x* to a single precision number.

CSRLIN Variable

Format: *v* = CSRLIN

Purpose: To obtain the vertical position of cursor.

CVI, CVS, CVD Functions

Format: *v* = CVI(*2-byte string*)
v = CVS(*4-byte string*)
v = CVD(*8-byte string*)

Purpose: To convert string values to numeric values.

DATA Statement

Format: DATA *constant* [, *constant*] ...

Purpose: To store the numeric and string constants that are accessed by the program's READ statement(s). (See READ Statement)

DATE$ Statement

Format: DATE$ = *x$*

x$ is a string expression indicated below.

mm-dd-yy
mm-dd-yyyy
mm/dd/yy
mm/dd/yyyy

Purpose: To set the current date. This statement complements the DATE$ function, which retrieves the current date.

DATE$ Variable

Format: v$ = DATE$

Purpose: To retrieve the current date.

DEF FN Statement

Format: DEF FN *name* [(*arg* [, *arg*]...)] = *function definition*

Purpose: To name define and name a user-defined function.

DEFINT/SNG/DBL/STR Statements

Format: DEF *type letter* [– *letter*][, *letter* [– *letter*]] ...

where *type* is INT, SNG, DBL, or STR

Purpose: To declare variable types as integer (DEFINT), single precision (DEFSNG), double precision (DEFDBL), or string (DEFSTR).

DEF SEG Statement

Format: DEF SEG [= *address*]

where *address* is a numeric expression in the range 0 to 65535.

Purpose: To assign the current segment address to be referenced by a subsequent BLOAD, BSAVE, CALL, or POKE statement, or by a USR or PEEK function.

DEF USR Statement

Format: DEF USR [*digit*] = *integer expression*

digit may be any digit from 0 to 9. *integer expression* is an integer expression in the range 0 to 65535.

Purpose: To specify the starting address of an assembly language subroutine.

DELETE Command

Format: DELETE [*line number*][–[*line number*]]

Purpose: To delete program lines.

DIM Statement

Format: DIM *variable* (*subscripts*)[, *variable* (*subscripts*)]...

Purpose: To specify the maximum values for array variable subscripts and allocate storage accordingly.

DRAW Statement

Format: DRAW *string expression*

where *string expression* is a set of subcommands.

Purpose: To draw an object defined by the subcommands described below.

Prefixes
The following prefix commands may precede any of the movement commands:

B Move but don't plot any points
N Move and return to original position when done

Cursor Movement
The following commands specify movement in units. The size of a unit may be modified by the S Command. If no argument is supplied, the cursor is moved one unit.

U [*n*] Move up
D [*n*] Move down
L [*n*] Move left
R [*n*] Move right
E [*n*] Move diagonally up and right
F [*n*] Move diagonally down and right
G [*n*] Move diagonally down and left
H [*n*] Move diagonally up and left

Other Commands
M *x, y* Move absolute or relative
A *n* Set angle *n*
TA *degrees* Rotate *degrees*
C *n* Set color *n*
S *n* Set scale factor
X *string expression*; Execute substring
P *paintcolor, bordercolor* Paint area enclosed with *bordercolor*

EDIT Command

Format: EDIT *line number*

Purpose: To display a specified line for editting

END Statement

Format: END

Purpose: To terminate program execution, close all files, and return to the BASIC command level.

ENVIRON Statement

Format: ENVIRON *string*

string must be of the form *parameter-id = text* or *parameter-id text.*

Purpose: To modify a parameter in the MS-DOS Environment String Table.

ENVIRON$ Function

Format: *v$* = ENVIRON$(*string parameter*)
*v$* = ENVIRON$(*n*)

where *n* is an integer.

Purpose: To retrieve a parameter string from BASIC's Environment String Table.

EOF Function

Format: *v* = EOF(*file number*)

Purpose: To test for the end-of-file.

ERASE Statement

Format: ERASE *array name* [, *array name*] ...

Purpose: To eliminate arrays from a program.

ERDEV,ERDEV$ Functions

Format: *v* = ERDEV
*v$* = ERDEV$

Purpose: To provide a way to obtain device-specific status information.

ERR and **ERL** Variables

Format: *v* = ERR
v = ERL

Purpose: To obtain the error code (ERR) and the line number (ERL) when an error occurred.

ERROR Statement

Format: ERROR *n*

where *n* is an integer expression in the range 0 to 255.

Purpose: To simulate the occurrence of a BASIC error, or to allow error codes to be defined by the user.

EXP Function

Format: *v* = EXP(*x*)

Purpose: To return e (base of natural logarithms) to the power of *x*, *x* must be <= 88.02968.

FIELD Statement

Format: FIELD [#] *file number* , *field width* AS *string variable* [, *field width* AS *string variable*] ...

Purpose: To allocate space for variables in a random file buffer.

FILES Statement

Format: FILES [*filespec*]

where *filespec* includes either a filename or a pathname and optional device designation.

Purpose: To print the names of files residing on the specified disk.

FIX Function

Format: *v* = FIX(*x*)

Purpose: To return the truncated integer part of *x*.

FOR...NEXT Statement

Format: FOR *variable* = *x TO y* [STEP *z*]

.
.
.

NEXT [*variable*][, *variable*] ...

where *x*, *y*, and *z* are numeric expressions.

Purpose: To perform the embedded loop a given number of times.

FRE Function

Format: *v* = FRE(*x*)
v = FRE(*x$*)

Purpose: To return the number of bytes in memory not being used by BASIC. The arguments within the parentheses are dummy arguments.

GET Statement - Files

Format: GET [#] *file number* [, *record number*]

Purpose: To read a record from a random disk file into a random buffer.

GET Statement - Graphics

Format: GET (*x1*, *y1*)–(*x2*, *y2*), *array name*

where (*x1*, *y1*)–(*x2*, *y2*) is a rectangular area on the display screen.
array name is the name assigned to the variable that will hold the graphic pattern.

Purpose: The GET and PUT Statements are used together to transfer graphic images to and from the screen.

GOSUB...RETURN Statements

Format: GOSUB *line number*

.
.
.

RETURN [*line number*]

Purpose: To branch to and then return from a subroutine.

GOTO Statement

Format: GOTO *line number*

Purpose: To branch unconditionally to a specified line number.

HEX$ Function

Format: *v$* = HEX$(*n*)

where *n* is a numeric expression in the range –32768 to 65535.

Purpose: To return a string that represents the hexadecimal value of the decimal within the parentheses.

IF...THEN[...ELSE]/IF...GOTO Statements

Format: IF *expression* [,] THEN { *statement(s)* | *line number* }
[,][ELSE { *statement(s)* | *line number* }]

IF *expression* [,] GOTO *line number*
[,][ELSE { *statement(s)* | *line number* }]

Purpose: To make a decision regarding program flow based on the result returned by an expression.

INKEY$ Function

Format: *v$* = INKEY$

Purpose: To return either a one-character string containing a character read from the standard input device or a null string if no character is pending there. The keyboard is usually the standard input device.

INP Function

Format: *v* = INP(*n*)

Purpose: To return the byte read from port *n*. *n* must be in the range 0 to 65535.

INPUT Statement

Format: INPUT [;][*"prompt string"* ;] *variable* [, *variable*] ...

Purpose: To allow input from the keyboard during program execution.

INPUT # Statement

Format: INPUT # *file number* , *variable* [, *variable*] ...

Purpose: To read data items from a sequential device or file and assign them to program variables.

INPUT$ Function

Format: *v$* = INPUT$(*n* [,[#] *file number*])

Purpose: To return a string of *n* characters, read from specified file. If the file number is not specified, the characters will be read from the standard input device.

INSTR Function

Format: *v* = INSTR([*n*,] *x$*, *y$*)

Purpose: To search for the first occurrence of string *y$* in *x$*, and to return the position at which the match is found. Optional offset *n* sets the position for starting the search.

INT Function

Format: *v* = INT(*x*)

Purpose: To return the largest integer <= *x*.

KEY Statement

Format:
KEY *n* , *x$*
KEY LIST
KEY ON
KEY OFF

n is the number of the function key.
x$ is the text assigned to the specified key.

Purpose: To assign values to function keys and display the values.

The default assignments are listed below:

F1 LIST	F2 RUN ←
F3 LOAD"	F4 SAVE"
F5 CONT ←	F6 ,"LPT1:" ←
F7 TRON ←	F8 TROFF ←
F9 KEY	F10 SCREEN 0,0,0 ←

The arrow (←) indicates ENTER.

KEY(*n*) Statement

Format:

KEY(*n*) ON
KEY(*n*) OFF
KEY(*n*) STOP

where *n* is the number of a function key,

1 through 10	F1 through F10
11 through 14	cursor control keys

Purpose: To enable or disable event trapping by a programmable-function key, a cursor direction key, or a user-defined.

KILL Command

Format: KILL [*filespec*]

Purpose: To delete a file or a pathname from disk.

LEFT$ Function

Format: *v$* = LEFT$(*x$*, *n*)

where *n* is a numeric expression in the range 0 to 255.

Purpose: To return a string of the leftmost *n* characters of *x$*.

LEN Function

Format: *v* = LEN(*x$*)

Purpose: To return the number of characters in *x$*.
Nonprinting characters and blanks are counted.

LET Statement

Format: [LET] *variable* = *expression*

Purpose: To assign the value of an expression to a variable. Notice that the word LET is optional; i.e., the equal sign is sufficient for assigning an expression to a variable name.

LINE Statement

Format: LINE [[STEP](*x1*, *y1*)]-[STEP](*x2*, *y2*)
[,[*color*][,[B [F]][, *style*]]]

(*x1*, *y1*) is the coordinate for the starting point of the line.
(*x2*, *y2*) is the ending point for the line.

Purpose: To draw a line or box on the screen.

B draws a box with the diagonal points (*x1*, *y1*) and (*x2*, *y2*).

BF draws a filled box.

LINE INPUT Statement

Format: LINE INPUT[;] [*"prompt string"* ;] *string variable*

Purpose: To assign an entire line (up to 254 characters) to a string variable without using delimiters.

LINE INPUT # Statement

Format: LINE INPUT # *file number* , *string variable*

Purpose: To assign an entire line (up to 254 characters), without delimiters, from a sequential disk data file to a string variable.

LIST Command

Format: LIST [*line number*][–[*line number*]][, *device*]

line number is in the range 0 to 65529.
device is a device designation string, such as SCRN: or LPT1:, or a filename.

Purpose: To list all or part of the program currently in memory.

LLIST Command

Format: LLIST [*line number*][–[*line number*]]

Purpose: To list all or part of the program currently in memory on the line printer.

LOAD Command

Format: LOAD *filespec* [, R]

Purpose: To load a file from an input device into memory. If R is specified, the program executes after it is loaded.

LOC Function

Format: *v* = LOC(*file number*)

where *file number* is the number under which the file was opened.

Purpose: With random disk files, LOC returns the record number within the file. With sequential files, LOC returns the current byte position in the file, divided by 128.

LOCATE Statement

Format: LOCATE [*row*][,[*col*][,[*cursor*][,[*start*][, *stop*]]]]

row is a line number (vertical) on the screen.
col is the column number on the screen.
cursor is a value indicating whether the cursor should be visible or not.
0: invisible
Non-zero: visible
start is the cursor starting line (vertical) on the screen, expressed numerically in the range of 0 to 31.
stop is the cursor stop line (vertical) on the screen, expressed numerically in the range of 0 to 31.

Purpose: Moves the cursor to the specified position. The optional parameter turns the cursor on/off and define the vertical start and stop lines.

LOF Function

Format: *v* = LOF(*file number*)

Purpose: To return the length of the specified file in bytes.

LOG Function

Format: *v* = LOG(*x*)

Purpose: To return the natural logarithm of *x*. *x* must be greater than zero.

LPOS Function

Format: *v* = LPOS(*n*)

n indicates which printer is being tested, as follows:

0 or 1	LPT1:
2	LPT2:
3	LPT3:

Purpose: To return the current position of the printer's print head within the printer buffer.

LPRINT and **LPRINT USING** Statements

Format: LPRINT [*list of expressions*][;]

LPRINT USING *x$* ; *list of expressions* [;]

x$ is a string expression which consists of formatting charactoers as described in the "PRINT USING Statement".

Purpose: To print data on the printer.

LSET and **RSET** Statements

Format: LSET *v$* = *x$*
RSET *v$* = *x$*

v$ is the name of a variable that was defined in a FIELD satement.

Purpose: To move data from memory to a random file buffer (in preparation for a PUT statement) or to left - or right - justify the value of a string into a string variable.

MERGE Command

Format: MERGE *filespec*

Purpose: To merge a specified ASCII program file into the program currently in memory.

MID$ Statement

Format: MID$(*v$*, *n* [, *m*]) = *x$*

where *n* is an integer expression in the range 1 to 255.
m is an integer expression in the range 0 to 255.
v$ is a string variable that will have its characters replaced.

Purpose: To replace a portion of one string variable with another string.

MID$ Function

Format: *v$* = MID$(*x$* , *n* [, *m*])

where *n* is an integer expression in the range 1 to 255.
m is an integer expression in the range 0 to 255.

Purpose: To return a string *m* characters in length from *x$*, beginning with the *n* th character.

MKDIR Statement

Format: MKDIR *pathname*

where *pathname* is a string expression specifying the name of the directory to be created.

Purpose: To create a new subdirectory.

MKI$, MKS$, MKD$ Functions

Format: *v$* = MKI$(*integer expression*)
*v$* = MKS$(*single precision expression*)
*v$* = MKD$(*double precision expression*)

Purpose: To convert numeric values to string values.

NAME Command

Format: NAME *old filename* AS *new filename*

Purpose: To change the name of a disk file.

NEW Command

Format: NEW

Purpose: To delete the program currently in memory and clear all variables.

OCT$ Function

Format: *v$* = OCT$(*n*)

where *n* is a numeric expression in the range –32768 to 655355.

Purpose: To return a string that represents the octal value of the decimal argument. *n* is rounded to an integer before OCT$ (*n*) is evaluated.

ON COM(*n*) Statement

Format: ON COM(*n*) GOSUB *line number*

where *line number* is the number of the first line of a subroutine.
n is the number of the communications port..

Purpose: To specify the first line number of a subroutine to be performed when activity occurs on a communications port.

ON ERROR GOTO Statement

Format: ON ERROR GOTO *line number*

where *line number* is the first line of the error trapping routine.

Purpose: To enable error handling and specify the first line of the error handling routine.

ON...GOSUB and ON...GOTO Statements

Format: ON *expression* GOTO *list of line numbers*
ON *expression* GOSUB *list of line numbers*

where *expression* is a numeric expression in the range 0 to 255.

Purpose: To branch to one of several specified line numbers depending on the value returned when an expression is evaluated.

ON KEY(*n*) Statement

Format: ON KEY(*n*) GOSUB *line number*

n is the number of a function key, or direction key (range 1 to 14).
line number is the number of the first line of a subroutine.

Purpose: To specify the first line number of a subroutine to be performed when a specified key is pressed.

ON PEN Statement

Format: ON PEN GOSUB *line number*

where *line number* is the number of the first line of a subroutine.

Purpose: To specify the first line number of a subroutine to be performed when the light pen is activated.

ON PLAY(*n*) Statement

Format: ON PLAY(*n*) GOSUB *line number*

n is an Integer expression in the range 1 through 32.
line number is the statement line number of the PLAY event trap subroutine.

Purpose: To branch to a specified subroutine when the music queue contains fewer than *n* notes.

ON STRIG(*n*) Statement

Format: ON STRIG *n* GOSUB *line number*

where *n* is the number of the joystick trigger.
0: button is A1 2: button is B1
4: button is A2 6: button is B2
line number is the number of the first line of a subroutine.

Purpose: To specify the first line number of a subroutine to be performed when the joystick trigger is pressed.

ON TIMER(*n*) Statement

Format: ON TIMER(*n*) GOSUB *line number*

n must be a numeric expression in the range of 1 to 86400 (1 second to 24 hours).
line number is the number of the first line of the ON TIMER event trap subroutine.

Purpose: To provide an event trap in real time.

OPEN Statement

Format: OPEN *mode1* ,[#] *file number* , *filespec* [, *record length*]

OPEN *filespec* [FOR *mode2*] AS [#] *file number* [LEN = *record length*]

Purpose: To allow I/O to a file or device.

Remarks *filespec* is an optional device specification followed by a filename or pathname, that conforms to the rules of MS-DOS 2.0 for filenames.

mode1 is a string expression. The first character must be one of the following:

"O"	Specifies sequential output mode
"I"	Specifies sequential input mode
"R"	Specifies random input/output mode
"A"	Specifies sequential append mode

mode2 is an expression which is one of the following:

OUTPUT	Specifies sequential output mode
INPUT	Specifies sequential input mode
APPEND	Specifies sequential append mode

file number is an integer expression between 1 and 255.

record length sets the record length for random files.

OPEN COM Statement

Format: OPEN "COMn : [*speed*][, [*parity*][, [*data*][, [*stop*] [, RS][, CS [n]][, DS [n]][, CD [n]][, BIN][, ASC][, LF]]]]" [FOR *mode*] AS [#] *filenumber* [LEN = *record length*]

Purpose: To open and initialize a communications channel for input/output.

Remarks COMn is a legal communications device, i.e., COM1: or COM2:.

speed is the baud rate, in bits per second, of the device to be opened.

parity designates the parity of the device to be opened. Valid entries are: N (none), E (even), O (odd), S (space), or M (mark).

data designates the number of data bits per byte. Valid entries are: 5, 6, 7, or 8.

stop designates the stop bit. Valid entries are 1, 1.5, or 2.

RS suppresses RTS (Request To Send)
CS[n] controls CTS (Clear To Send)
DS[n] controls DSR (Data Set Ready)
CD[n] controls CD (Carrier Detect)

BIN opens the device in binary mode. BIN is selected by default unless ASC is specified.
ASC opens the device in ASCII mode.

LF specifies that a line feed is to be sent after a carriage return.

mode is one of the following string expressions:
OUTPUT specifies sequential output mode.
INPUT specifies sequential input mode.
If the *mode* is omitted, random input/output mode is chosen.

file number is the number of the file to be opened.

OPTION BASE Statement

Format: OPTION BASE *n*

where *n* is 1 or 0

Purpose: To declare the minimum value for array subscripts.

OUT Statement

Format: OUT *n* , *m*

where *n* is the port number in the range 0 to 65535. *m* is the data to be transmitted in the range 0 to 255.

Purpose: To send a byte to an output port.

PAINT Statement

Format: PAINT (*x start* , *y start*) [, *paint attribute* [, *border color*] [, *background attribute*]]

x start and *y start* are the coordinates where painting is to begin.
paint attribute is a numeric expression that indicates the color to be painted.
border color identifies the border color of the figure to be filled.
background attribute is a string formula returning character data. When it is omitted, the default is CHR$(0).

Purpose: To fill a graphic area with the color or pattern specified.

PEEK Function

Format: *v* = PEEK(*n*)

n is an integer in the range 0 to 65535.

Purpose: To return the byte read from the indicated memory location *n*. PEEK is the complementary function of the POKE Statement.

PEN ON, PEN OFF, PEN STOP Statements

Format: PEN ON
PEN OFF
PEN STOP

Purpose: The PEN ON enables the light pen.
The PEN OFF disables the light pen read.
The PEN STOP suspends the light pen trapping.

PEN Function

Format: *v* = PEN(*n*)

Purpose: To read the light pen

Remarks: The *v* = PEN(*n*) function reads the light pen coordinates, where *n* is:

0 – If the pen was down since the last poll
1 – Returns the *x* pixel coordinate
2 – Returns the *y* pixel coordinate
3 – Returns the current pen switch value
4 – Returns the last known valid *x* pixel
5 – Returns the last known valid *y* pixel
6 – Returns the character row position
7 – Returns the character column position
8 – Returns the last known character row
9 – Returns the last known valid character

PLAY Statement

Format: PLAY *string*

string is one or more of the subcommands listed under "Remarks".

Purpose: To play music as specified by *string*.

Remarks: A set of subcommands, used as part of the PLAY Statement, specifies the particular action to be taken.

Change Octave

< increments octave.
> decrements octave.

Tone

O *n* Sets the current octave.
A–G Plays a note in the range A-G.
N *n* Plays note n.

Duration

L *n* Sets the length of each note.
MN Sets "music normal".
ML Sets "music legato".
MS Sets "music staccato".

Tempo

P *n* Specifies a pause.
T *n* Sets the "tempo".

Operation

MF Sets music (PLAY Statement) and SOUND to run in the foreground.
MB Music (PLAY Statement) and SOUND are set to run in the background.

Substring

X *x$* ; Executes a substring.

Suffixes

or + Follows a specified note, and turns it into a sharp.
– Follows a specified note, and turns it into a flat.
. A period after a note causes the note to play 3/2 times the length determined by L multiplied by T.

PLAY Function

Format: *v* = PLAY(*n*)

n is a dummy argument and may be any value.

Purpose: To return the number of notes currently in the background music queue. PLAY(*n*) will return 0 if the user is in Music Foreground Mode.

PLAY ON, PLAY OFF, PLAY STOP Statements

Format:
PLAY ON
PLAY OFF
PLAY STOP

Purpose:
PLAY ON enables play event trapping.
PLAY OFF disables play event trapping.
PLAY STOP suspends play event trapping.

PMAP Function

Format: *v* = PMAP(*expression* , *function*)

Purpose: To map world coordinate expressions to physical locations or to map physical expressions to a world coordinate location.

Remarks: *function*:

0 Maps world expression to physical *x* coordinate
1 Maps world expression to physical *y* coordinate
2 Maps physical expression to world *x* coordinate
3 Maps physical expression to world *y* coordinate

POINT Function

Format: *v* = POINT(*x coodinate* , *y coodinate*)

x coodinate and *y coodinate* are the coordinates of the pixel that is to be referenced.

v = POINT(*function*)

Purpose: POINT(*x coodinate* , *y coodinate*) allows the user to read the color number of a pixel from the screen. If the specified point is out of range, the value –1 is returned.

v = POINT(*function*) returns the value of the current *x* or *y* Graphics accumulator as follows:

Remarks:

0 Returns the current physical *x* coodinate.
1 Returns the current physical *y* coodinate.
2 Returns the current logical *x* coodinate.
3 Returns the current logical *y* coodinate.

POKE Statement

Format: POKE *n* , *m*

where *n* and *m* are integer expressions.

Purpose: To write a byte into the specified memory location.

POS Function

Format: *v* = POS(*n*)

n is a dummy argument.

Purpose: To return the current horizontal (column) position of the cursor.

PRESET Statement

Format: PRESET [STEP] (*x coodinate* , *y coodinate*) [, *color*]

x coodinate and *y coodinate* are coordinates that specify the pixel to be set.
STEP indicates the given *x* and *y* coodinates will be relative.

Purpose: To draw a specified point on the screen. PRESET works like PSET except that if *color* is not specified, the background color is selected.

PRINT Statement

Format: PRINT [*list of expressions*]

where *list of expression* is a list of expressions separated by commas, blanks or semicolons.

Purpose: To output data to the screen.

PRINT USING Statement

Format: PRINT USING *x$* ; *list of expressions* [;]

x$ is a string using special formatting characters. The formatting characters (listed below) set the field and the format of the printed strings or numbers.
list of expressions consists of the string expressions or numeric expressions that are to be printed, separated by semicolons.

Purpose: To output strings or numbers to the screen using aspecified format.

String Field

"!"	Specifies the use of only the first character in the given string.
"\n spaces\"	Specifies that 2+n characters from the string are to be printed.
"&"	Specifies a variable length string field.

Numeric Fields

"#"	A number sign is used to represent each digit position.
.	A decimal point may be inserted at any position in the field.
+	A plus sign at the beginning or end of the format string will cause the sign of the number (plus or minus) to be printed before or after the number.
–	A minus sign at the end of the format field will cause negative numbers to be printed with a trailing minus sign.
**	A double asterisk at the beginning of the format string causes leading spaces in the numeric field to be filled with asterisks.

$$	A double dollar sign causes a dollar sign to be printed immediately to the left of the formatted number.
**$	The **$ at the beginning of a format string combines the effects of the above two symbols.
,	A comma to the left of the decimal point in a formatting string causes a comma to be printed to the left of every third digit from the decimal point.
^^^^	Four carets (or up-arrows) may be placed after the digit position characters to specify exponential format. The four carets specify E xx to be printed.
_	An underscore in the format string causes the next character to be output as a literal character.
%	If the number to be printed is larger than the specified numeric field, a percent sign is printed in front of the number.

PRINT# and **PRINT# USING** Statements

Format: PRINT# *file number* ,[USING *x$* ;] *list of expressions*

file number is the number used when the file was opened for output.
x$ is a string expression which consists of formatting characters as described in "PRINT USING Statement".
list of expressions is the numeric and/or string expression that will be written to the file.

Purpose: To write data to a sequential file.

PSET Statement

Format: PSET [STEP] (*x coordinate* , *y coordinate*) [, *color*]

where *x coordinate* and *y coordinate* represents the coordinates of the point to be set.
color is the number of the color to be used.

STEP indicates the given *x* and *y* coordinates will be relative, not absolute. That means the *x* and y are distances from the most recent cursor location, not distances from the (0,0) screen coordinate.

Purpose: To draw a specified point on the screen.

PUT Statement - Files

Format: PUT [#] *file number* [, *record number*]

file number is the number under which the file was opened.

record number is the number of records in the range 1 to 32767.

Purpose: To write a record from a random buffer to a random access file.

PUT Statement - Graphics

Format: PUT (*x1* , *y1*), *array name* [, *action verb*]

(*x1* , *y1*) in the PUT Statement specifies the point where a stored image is to be displayed.
array name is the name of array assigned to the place that voice hold the image.
action verb is one of: PSET, PRESET, AND, OR, XOR.

Purpose: The GET and PUT Statements are used together to transfer graphic images to and from the screen.

RANDOMIZE Statement

Format: RANDOMIZE [*n*]

where *n* is an integer expression.

Purpose: To reseed the random number generator.

READ Statement

Format: READ *variable* [, *variable*] ...

Purpose: To read values from a DATA Statement and assign them to variables. (See DATA Statement.)

REM Statement

Format: REM [*remark*]

where remark may be any character string.

Purpose: To allow explanatory remarks to be inserted in a program.

RENUM Command

Format: RENUM [[*new number*][,[*old number*][, *increment*]]]

where *new number* is the first line number to be used in the new sequence. the default is 10.
old number is the line in the current program where renumbering is to bigin. The default is the first line of the program.
increment is the increment to be used in the new sequence. The default is 10.

Purpose: To renumber program lines.

RESET Command

Format: RESET

Purpose: To close all files.

RESTORE Statement

Format: RESTORE [*line number*]

where *line number* is the line number of a statement in a program.

Purpose: To allow DATA Statements to be reread from a specified line.

RESUME Statement

Format: RESUME
RESUME 0
RESUME NEXT
RESUME *line number*

Purpose: To continue program execution after an error recovery procedure has been performed.

RETURN Statement

Format: RETURN [*line number*]

where *line number* is the line of the program to return.

Purpose: To return from a subroutine.

RIGHT$ Function

Format: *v$* = RIGHT$(*x$*, *n*)

where *x$* is any string expression.
n is any integer expresion.

Purpose: To return the rightmost *n* characters of string *x$*.

RMDIR Statement

Format: RMDIR *pathname*

where *pathname* is the name of directory to be deleted.

Purpose: To remove an existing directory.

RND Function

Format: *v* = RND[(*x*)]

Purpose: To return a random number between 0 and 1.

RUN Statement/Command

Format: RUN [*line number*]
RUN *filespec* [,R]

where *line number* is the line munber of the program in memory where execution begins.

Purpose: To execute the program currently in memory, or to load a file into memory and run it.

SAVE Command

Format: SAVE *filespec* [, { A | P }]

Purpose: To save a program file on disk.

SCREEN Statement

Format: SCREEN *mode* [,[*burst*][, *apage* [, *vpage*]]]

Purpose: To set the screen mode, optionally set the active (apage)/display (vpage) page.

Remarks *mode* : numerical expression indicating graphic resolution

0: text mode
1: low resolution graphics (320 * 200, color)
2: high resolution graphics (640 * 200, monocrome)

burst : numerical expression sets display color (*burst* = 0) or monocrome (*burst* = non-zero).

apage and *vpage* are valid in text mode only.

apage : numerical expression with the value of 0 to 7 for width 80, or 0 to 3 for width 40 to select active page.

vpage : numerical expression with the value of 0 to 7 for width 80, or 0 to 3 for width 40 to select display page.

SCREEN Function

Format: *v* = SCREEN(*row*, *column* [, *z*])

where *row* is a expression returning in the range 1 to 25. *column* is a numeric expression in the range 1 to 40 or 1 to 80 depending on the WIDTH setting. *z* is a valid numeric expression.

Purpose: To read a character or its color from a specified screen location. If the optional parameter *z* is given and is non-zero, the current color attribute for the screen is returned instead.

SGN Function

Format: *v* = SGN(*x*)

where *x* is any numeric expression.

Purpose: To indicate the value of *x*, relative to zero:

If *x* >0, SGN(*x*) returns 1.
If *x* = 0, SGN(*x*) returns 0.
If *x* <0, SGN(*x*) returns –1.

SHELL Statement

Format: SHELL [*command-string*]

where *command-string* is a string expression containing the name of a program to run and optionally command arguments.

Purpose: To exit the BASIC program, run a COM or EXE or BAT program, or a built in DOS command such as DIR or TYPE, and return to the BASIC program at after execution of the SHELL Statement.

SIN Function

Format: *v* = SIN(*x*)

Purpose: To return the sine of *x*, where *x* is in radians.

SOUND Statement

Format: SOUND *freq*, *duration*

freq is the desired frequency in Hertz. A numeric expression in the range 37 to 32767.
duration is the duration in clock ticks. Clock ticks occur 18.2 times per second. This must be a numeric expression in the range 0 to 65535.

Purpose: To generate a sound through the speaker.

SPACE$ Function

Format: *v$* = SPACE$(*n*)

where *n* must be in the range 0 to 255.

Purpose: To return a string of spaces of length *n*.

SPC Function

Format: *v* = SPC(*n*)

where *n* must be in the range 0 to 255.

Purpose: To skip spaces in a PRINT Statement. *n* is the number of spaces to be skipped.

SQR Function

Format: *v* = SQR(*x*)

where *x* must be greater than or equal to zero.

Purpose: To return the square root of *x* .

STICK Function

Format: *v* = STICK(*n*)

where *n* is a numeric expression returning an unsigned integer in the range 0 to 3.

Purpose: To return the *x* and *y* coordinates of the two joysticks.

Remarks The values returned for *n* can be:

0 – returns the *x* coordinate for joystick A. Also stores the *x* and *y* values for both joysticks for the following function calls:
1 – Returns the *y* coordinate of joystick A.
2 – Returns the *x* coordinate of joystick B.
3 – Returns the *y* coordinate of joystick B.

STOP Statement

Format: STOP

Purpose: To terminate program execution and return to command level.

STR$ Function

Format: *v$* = STR$(*x*)

where *x* is a numeric expression.

Purpose: To return the string equivalent of the value of *x* .

STRIG(*n*) Statements

Format:

STRIG(*n*) ON
STRIG(*n*) OFF
STRIG(*n*) STOP

where *n* is 0, 2, 4 or 6, and indicating the buttons (triggers) to be trapped as follows:

0 button A1, 2 button B1
4 button A2, 6 button B2

Purpose:

The STRIG(*n*) ON statement enables joystick trapping.
The STRIG(*n*) OFF statement disables joystick trapping.
The STRIG(*n*) STOP statement suspends joystick trapping.

STRIG(*n*) Function

Format:

v = STRIG(*n*)

where *n* is a numeric expression in the range 0 to 7, designating whitch trigger is to be checked.

Purpose:

To return the status of a specified joystick trigger.

In the STRIG(*n*) function, the values returned for (*n*) can be:

0 – Returns –1 if button A1 was pressed since the last STRIG(0) function call, returns 0 if not.
1 – Returns –1 if button A1 is currently down, 0 if not.
2 – Returns –1 if button B1 was pressed since the last STRIG(2) function call, returns 0 if not.
3 – Returns –1 if button B1 is currently down, 0 if not.
4 – Returns –1 if button A2 was pressed since the last STRIG(4) function call, returns 0 if not.
5 – Returns –1 if button A2 is currently down, 0 if not.
6 – Returns –1 if button B2 was pressed since the last STRIG(6) function call, returns 0 if not.
7 – Returns –1 if button B2 is currently down, 0 if not.

STRING$ Function

Format: *v$* = STRING$(*n* , *m*)
*v$* = STRING$(*n* , *x$*)

where *n* and *m* are in the range 0 to 255.

Purpose: To return a string of length *n* whose characters all have ASCII code *m* or the first character of *x$*.

SWAP Statement

Format: SWAP *variable* , *variable*

Purpose: To exchange the values of two variables.

SYSTEM Command

Format: SYSTEM

Purpose: To close all open files and return control to the operating system.

TAB Function

Format: TAB(*n*)

where *n* must be in the range 1 to 255.

Purpose: To move the print position to *n*.

TAN Function

Format: *v* = TAN(*x*)

Purpose: To return the tangent of *x*. *x* should be given in radians.

TIME$ Statement

Format: TIME$ = *x$*

where *x$* is a string expression indicating the time to be set as explained bellow.

x$ returns a string in one of the following forms:

hh (sets the hour)
hh:mm (sets the hour and minutes)
hh:mm:ss (sets the hour, minutes, and seconds)

Purpose: To set the time. This statement complements the TIME$ Function, which retrieves the time.

TIME$ Variable

Format: *v$* = TIME$

Purpose: To retrieve the current time. (To set the time, use the TIME$ Statement.)

TIMER Statements

Format: TIMER ON
TIMER OFF
TIMER STOP

Purpose: TIMER ON enables real time trapping.
TIMER OFF disables real time trapping.
TIMER STOP suspends real time trapping.

TRON, TROFF Commands

Format: TRON
TROFF

Purpose: To trace the execution of program statements.

Remarks As an aid in debugging, the TRON Statement may be executed in either direct or indirect mode. With TRON in operation, each line number of the program is printed on the screen as it is executed.

USR Function

Format: *v* = USR[*digit*][(*argument*)]

where *digit* is in the range 0 to 9 specifies which USR routine is being called.
argument is the value passed to the subroutine. It may be any numeric or string expression.

Purpose: To call an assembly language subroutine.

VAL Function

Format: *v* = VAL(*x$*)

x$ must be a numeric character stored as a string.

Purpose: To return the numeric value of string *x$*. The VAL Function also strips leading blanks, tabs, and feeds a line from the argument string.

VARPTR Function

Format: *v* = VARPTR(*variable name*)
v = VARPTR(#*file number*)

Purpose: Returns the address of the first byte of data identified with *variable name*. The variable must have been defined prior to the execution of the VARPTR Function.

For sequential files, returns the starting address of the disk I/O buffer assigned to *file number*. For random files, returns the address of the FIELD buffer assigned to *file number*.

VARPTR$ Function

Format: *v$* = VARPTR$(*variable name*)

where *variable name* is the name of a variable in the program.

Purpose: To return a character form of the memory address of the variable in a string form.

VIEW Statement

Format: VIEW [[SCREEN][(*Vx1* , *Vy1*)–(*Vx2* , *Vy2*)[,[*color*] [,[*border*]]]]]

where (*Vx1* , *Vy1*) is upper left *x* , *y* coodinates. (*Vx2* , *Vy2*) is lower right *x* , *y* coodinates.

SCREEN option dicates that the *x* and *y* coodinates are absolute to the screen, not relative to the border of the physical viewport, and only graphics within the viewport will be plotted.

color attribute allows the user to fill the view area with a color.

border attribute allows the user to draw a line surrounding the viewport.

Purpose: To define screen limits for graphics activity.

VIEW PRINT Statement

Format: VIEW PRINT [*top screen line* TO *bottom screen line*]

Purpose: To set the boundaries of the screen text window.

WAIT Statement

Format: WAIT *port number* , *n* [, *m*]

where *n* and *m* are integer expressions in the range 0 to 255.
port number is the port number in the range 0 to 65535.

Purpose: To suspend program execution while monitoring the status of a machine input port.

WHILE...WEND Statements

Format:

WHILE *expression*
.
.
[*loop statements*]
.
.
WEND

Purpose: To execute a series of statements in a loop as long as a given condition is true.

WIDTH Statement

Format:

WIDTH *size* [, *row*]
WIDTH *file number* , *size*
WIDTH *device* , *size*

size is a numeric expression in the range 0 to 255.
file number is an integer within the range of 1 to 15. This is the number of the file that is open.
device is a valid string expression returning the device identifier. Valid devices are SCRN: , LPT1:, LPT2: and LPT3:.

Purpose: To set the printed line width in number of characters for the screen, line printer or other devices.

WINDOW Statement

Format: WINDOW [[SCREEN](*Wx1* , *Wy1*) – (*Wx2* , *Wy2*)]

(*Wx1* , *Wy1*) – (*Wx2* , *Wy2*) are the world coordinates specified the to define the coordinates of the lower left and upper right screen border.

SCREEN inverts the *y* axis of the world coodinates so that screen coordinates coincide with the traditional Cartesian arrangement: *x* increases left to right, and *y* decreases top to bottom.

Purpose: To define the logical dimensions of the current viewport.

WRITE Statement

Format: WRITE [*list of expressions*]

Purpose: To output data to the screen.

WRITE# Statement

Format: WRITE# *file number* , *list of expressions*

where *file number* is the number under which the file was opend for output.

Purpose: To write data to a sequential file.

WRITE# inserts commas between the items as they are written to the file and delimits strings with quotation marks.

CHAPTER 4.
MS-DOS INSTRUCTION

Introduction 4-2
- What is MS-DOS? 4-2
- Files 4-2
  - What is a file? 4-2
  - DIR (Directory) command 4-2
  - CHKDSK (Check Disk) command 4-3

More about Files 4-4
- How to name your files 4-4
- Wild cards 4-5
  - ? wild card 4-5
  - * wild card 4-5
- Illegal filenames 4-7
- Directories 4-7
- Filenames and paths 4-8
  - Pathnames 4-8
  - Pathing and external commands 4-8
  - Creating a directory 4-9
  - How to change the working directory 4-9
  - How to remove a directory 4-9

Learning about Commands 4-10
- Types of MS-DOS commands 4-10
- Information common to all MS-DOS commands 4-10
- Batch processing 4-11
- AUTOEXEC.BAT file 4-12
- Input and output 4-12
  - Redirecting output 4-12
  - Filters 4-13
  - Command piping 4-13

Introduction

What is MS-DOS?

Microsoft(R) MS(tm)-DOS is a disk operating system for 8086/8088-based computers. Using MS-DOS, you can control the system resources (the computer, disk drives, and printer).

The operating system is the program that provides the interface between the computer system's hardware and its software applications. It allows you to directly control the hardware.

MS-DOS is such a disk operating system. It enables you to create and keep track of files, run and link programs, and access peripheral devices (printers and disk drives) that are attached to your computer. MS-DOS represents an important advance in microprocessor operating systems.

Files

What is a file?
A file is a unit of information. A file on a disk can be compared to a file folder in a desk drawer. For example, one file folder might contain the names and addresses of the employees who work in the office. You might name this file the Employee Master File. A file on a disk could also contain the names and addresses of employees in the office and could be named Employee Master File.

All programs, text, and data on a disk must be placed in files. Each file has a unique name. Files are referenced by their names. "More About Files" tells you how to name your files.

DIR (Directory) command
If you want to know what files are on your disk, use the DIR command. This command tells MS-DOS to display all the files in the current directory on the disk that is named. For example, if your MS-DOS disk is in drive A: and you want to see the listing for the current directory on that disk, type:

```
DIR A:
```

MS-DOS will respond with a list of all the files in the current directory on your MS-DOS disk.

Note that Two MS-DOS system files, IO.SYS and MSDOS.SYS, are "hidden" files and will not appear when you issue the DIR command.

You can also get information about any file on your disk by typing DIR and a filename. For example, if you have created a file named MYFILE.TXT, the following command will display all directory information (name of file, its size, date last edited) for the file MYFILE.TXT.

```
DIR MYFILE.TXT
```

CHKDSK (Check disk) command

The MS-DOS command CHKDSK is used to check your disks for the amount of free space.

CHKDSK analyzes the directories and the File Allocation Table on the disk that you specify. It then produces a status report of any inconsistencies, such as files which have a non-zero size in their directory but really have no data in them.

To check the disk in drive A:, type the following command:

```
CHKDSK A:
```

MS-DOS will display a status report and any errors that it has found. An example of this display and more information on CHKDSK can be found in the description of the CHKDSK command in Chapter 5. You should run CHKDSK occasionally to ensure the integrity of your files.

More about Files

How to name your files

A typical MS-DOS file is named like this:

NEWFILE.TXT

A filename consists of two parts. The filename is NEWFILE and the filename extension is .TXT.

A filename can be from 1 to 8 characters long. The filename extension can be three or fewer characters, or may be omitted. The filename may be entered in small or capital letters. MS-DOS will translate these letters into uppercase characters.

In addition to the filename and the filename extension, you may designate a drive. The drive designation tells MS-DOS to look on the disk in the designated drive to find the filename. For example, to find directory information about the file NEWFILE.EXE on the disk in drive A: (when drive A: is NOT the default drive), type the following command:

DIR A:NEWFILE.EXE

Directory information about the file NEWFILE.EXE is now displayed on your screen.

Your filenames will probably be made up of letters and numbers, but other characters are also allowed. Here is a complete list of the characters you can use in filenames and extensions:

A–Z 0–9 $ &

% ' ( ) –

@ ^ { } ~ !

All parts of a filename comprise a file specification. The term file specification (or filespec) will be used in this manual to indicate the following filename

format:

[*drive designation* :] *filename* [*.filename extension*]

Examples of file specifications are:

B:MYPROG.COB
A:YOURPROG.EXT
A:NEWFILE.
TEXT

Wild cards

Two special characters (called "wild cards") can be used in filenames and extensions: the asterisk (*) and the question mark (?). These special characters give you greater flexibility when specifying filenames in MS-DOS commands.

? wild card
A question mark (?) in a filename or filename extension indicates that any character can occupy that position. For example, the following MS-DOS command will list all directory entries on the default drive that have 8 characters, begin with TEST, have any next character, end with the letters RUN, and have a filename extension of .EXE:

DIR TEST?RUN.EXE

Here are some examples of files that might be listed by the above DIR command:

TEST1RUN.EXE
TEST2RUN.EXE
TEST6RUN.EXE

*** wild card**
An asterisk (*) in a filename or filename extension indicates that any character can occupy that position or any of the remaining positions in the filename or extension. For example, the following command will list all directory entries on the default drive with filenames that begin with the characters TEST and have an extension of .EXE:

DIR TEST*.EXE

Here are some examples of files that might be listed by the above DIR command:

TEST1RUN.EXE
TEST2RUN.EXE
TEST6RUN.EXE
TESTALL.EXE

The wild card designation *.* refers to all files on the disk. Note that this can be very powerful and destructive when used in MS-DOS commands. For example, the command DEL *.* deletes all files on the default drive, regardless of filename or extension.

Examples:

To list the directory entries for all files named NEWFILE on drive A: (regardless of their filename extensions), simply type:

DIR A:NEWFILE.*

To list the directory entries for all files with filename extensions of .TXT (regardless of their filenames) on the disk in drive B:, type:

DIR B:????????.TXT or DIR B:*.TXT

This command is useful if, for example, you have given all your text programs a filename extension of .TXT. By using the DIR command with the wild card characters, you can obtain a list of all your text files even if you do not remember all of their filenames.

Illegal filenames

MS-DOS treats some device names specially, and certain 3-letter names are reserved for the names of these devices. These 3-letter names cannot be used as filenames or extensions. You must not name your files any of the following:

AUX Used when referring to input from or output to an auxiliary device (such as a printer or disk drive).

CON Used when referring to keyboard input or to output to the terminal console (screen).

LST or
PRN Used when referring to the printer device.

NUL Used when you do not want to create a particular file, but the command requires an input or output filename.

Directories

The names of your files are kept in a directory on each disk. The directory also contains information on the file sizes, their locations on the disk, and the dates that they were created or updated.

MS-DOS allows you to organize the files on your disks into directories. Directories are a way of dividing all your files into convenient groups. A directory can contain any number of files, and it may also contain other directories (referred to as subdirectories). This is called a hierarchical directory structure.

This can be thought to have a "tree" structure: directories are branches of the tree and files are the leaves, except that the "tree" grows downward; that is, the "root" is at the top. (Refer to Chapter 2 Pathnames.)

The filenames discussed earlier in this chapter are relative to your current directory. When you turn on your computer, you are "in" the root directory. Unless you take special action when you create a file, the new file is created in the directory in which you are now working. Files can have the same name if they are located on different directories.

Filenames and paths

If you use hierarchical directories, you must tell MS-DOS where the files are located in the directory structure. This is done by giving MS-DOS a pathname to the file.

Pathnames

A simple filename is a file specification described above. A pathname is a sequence of directory names followed by a simple filename, each separated from the previous one by a backslash (\ ).

The syntax of pathnames is:

[*d* :][.\ ][*directory*] \ [*directory...*] \ [*filename*]

If a pathname begins with a bacslash, MS-DOS searches for the file beginning at the root (or top) of the tree. Otherwise, MS-DOS begins at the user's current directory, known as the working directory, and searches downward.

When you are in your working directory, a filename and its corresponding pathname may be used interchangeably. Some sample names are:

\	Indicates the root directory.
\PROGRAMS	Sample directory under the root directory containing program files.
USER\SUE	A relative pathname; it specifies the file or directory SUE in subdirectory USER of the working directory. If the working directory is the root (\), it specifies \USER\SUE.

Pathing and external commands

External commands reside on disks as program files. They must be read from the disk before they execute.

When you are working with more than one directory, it is convenient to put all MS-DOS external commands into a separate directory. You must tell MS-DOS the name of the directory that contains these external commands. Do this with the PATH command.

Creating a directory
To create a subdirectory in your working directory, use the MKDIR (Make Directory) command. For example, to create a new directory named NEWDIR under your working directory, simply type the following command:

MKDIR NEWDIR (or MD NEWDIR)

After this command has been executed by MS-DOS, your working directory will contain a new subdirectory. You can also make directories anywhere in the tree structure by specifying MKDIR and then a pathname. MS-DOS will automatically create the "." and ".." entries in the new directory.

How to change the working directory
Changing from your working directory to another directory is very easy in MS-DOS. Simply issue the CHDIR (Change Directory) command and supply a pathname. For example:

CHDIR \USER (or CD \USER)

changes the working directory from \USER\JOE to \USER. You can specify any pathname after the command to "travel" to different branches and leaves of the directory tree.

How to remove a directory
To delete a directory in the tree structure, use the MS-DOS RMDIR (Remove Directory) command. For example, to remove the directory NEWDIR from the working directory, type:

RMDIR NEWDIR (or RD NEWDIR)

Note that the directory NEWDIR must be empty except for the "." and ".." entries before it can be removed; this will prevent you from accidentally deleting files and directories. You can remove any directory by specifying its pathname. To remove the \BIN\USER\JOE directory, make sure that it has only the "." and ".." entries, then type:

RMDIR \BIN\USER\JOE

Learning about Commands

Types of MS-DOS commands

There are two types of MS-DOS commands:

Internal commands
External commands

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you do a directory listing on your MS-DOS disk; they are part of the command processor. When you type these commands, they execute immediately.

External commands reside on disks as program files. They must be read from the disk before they can execute. If a disk containing the command is not in the drive, MS-DOS will not be able to find and execute the command.

Any filename with a filename extension of .COM, .EXE or .BAT is considered an external command. For example, programs such as FORMAT.COM and CHKDSK.COM are external commands. When you enter an external command, do not include its filename extension.

Information common to all MS-DOS commands

The following information applies to all MS-DOS commands:

1. Commands are usually followed by one or more options.

2. Commands and options must be separated by delimiters. Usually the space and comma are used as delimiters because they are the easiest. In this manual, we will use a space as the delimiter in commands.

3. Do not separate file specifications with delimiters, since the colon and the period already serve as delimiters.

4. You must include the filename extension when referring to a file that has a filename extension unless it is a system command file, in which case you must eliminate the extension.

5. You can abort commands when they are running by pressing CONTROL-BREAK .

6. Commands are activated only after you press the ENTER key.

7. Wild cards and device names (for example, PRN or CON) are not allowed in command names.

8. When commands produce a large amount of output on the screen, the display will automatically scroll to the next screen. You can press Ctrl-S to suspend the display. Press any key to resume the scrolling.

9. MS-DOS editing and function keys can be used when entering commands.

Batch processing

Often you may find yourself typing the same sequence of commands over and over to perform some commonly used task. With MS-DOS, you can put the command sequence into a special file called a batch file, and execute the entire sequence simply by typing the name of the batch file. "Batches" of commands in such files are processed as if they were typed on the keyboard. Each batch file must have the .BAT extension, and is executed by typing the filename without its extension.

You can create a batch file by using the Line Editor (EDLIN) or by using the COPY command (i.e. COPY CON XYZ.BAT).

Batch processing is useful if you want to execute several MS-DOS commands with one batch command, such as when you format and check a new disk. For example, a batch file for this purpose might look like this:

```
REM  This is a file to check new disks
REM  It is named NEWDISK.BAT
PAUSE Insert new disk in drive B:
FORMAT B:
DIR B:
CHKDSK B:
```

To execute this .BAT file, simply type the filename without the .BAT extension:

```
NEWDISK
```

The result is the same as if each of the lines in the .BAT file was entered as individual commands.

AUTOEXEC.BAT file

An AUTOEXEC.BAT file allows you to automatically execute programs when you start MS-DOS. Automatic Program Execution is useful when you want to run a specific program (for example, Microsoft Multiplan) everytime you start your computer.

Input and output

MS-DOS always assumes that input comes from the keyboard and output goes to the screen. However, the flow of command input and output can be redirected. Input can come from a file rather than a keyboard, and output can go to a file or to a line printer instead of to the screen. In addition, "pipes" can be created that allow output from one command to become the input to another.

Redirecting output

Most commands produce output that is sent to screen. You can send this information to a file by using a greater-than sign (>) in your command.

For example, DIR command displays a directory listing of the disk in the default drive. The same command can send this output to a file named MYFILES by designating the output file on the command line:

```
DIR >MYFILES
```

If the file MYFILES does not already exist, MS-DOS creates it and stores your directory list in it. If MYFILES already exists, MS-DOS overwrites what is in the file with the new data.

It is often useful to have input for a command come from a file rather than from a keybord. This is possible in MS-DOS by using a less-than sign (<) in your command. For example, the following command sorts the file NAMES and sends the sorted output to a file named LIST1:

```
SORT <NAMES >LIST1
```

Filters
A filter is a program that reads your input, transforms it in some way, and then outputs it, usually to your screen or to a file. In this way, the data is said to have been "filtered" by the program.

MS-DOS filters include FIND, MORE, and SORT. Their functions are described below:

FIND	Searches for a constant string of text in a file
MORE	Takes standard terminal output and displays it, one screen at a time
SORT	Sorts text

Command piping
If you want to give more than one command to the system at a time, you can "pipe" commands to MS-DOS. For example, you may occasionally need to have the output of one program sent as the input to another program.

Piping is done by separating commands with the pipe separator, which is the vertical bar symbol (¦). For example, the command

```
DIR  ¦ SORT
```

will give you an alphabetically sorted listing of your directory. The vertical bar causes all output generated by the left side of the bar to be sent to the right side of the bar for processing.

Piping can also be used when you want to output to a file. If you want your directory sorted and sent to a new file (for example, DIREC.FIL), you could type:

```
DIR ¦ SORT >DIREC.FIL
```

A pipeline may consist of more than two commands. For example,

```
DIR ¦ SORT ¦ MORE
```

will sort your directory, show it to you one screen at a time, and put —MORE— at the bottom of your screen when there is more output to be seen.

CHAPTER 5.
MS-DOS COMMANDS

MS-DOS Commands 5-3
- BREAK 5-5
- CD 5-5
- CHKDSK 5-6
- CLS 5-6
- COPY 5-7
- CTTY 5-8
- DATE 5-8
- DEL 5-9
- DIR 5-9
- DISKCOPY 5-10
- EXIT 5-11
- FC 5-11
- FIND 5-13
- FORMAT 5-14
- MD 5-14
- MORE 5-15
- PATH 5-15
- PRINT 5-16
- PROMPT 5-17
- RD 5-18
- RECOVER 5-18
- REN 5-19
- SET 5-19
- SORT 5-20
- SYS 5-20
- TIME 5-21
- TYPE 5-21
- VER 5-22
- VERIFY 5-22
- VOL 5-22

Batch Commands (Command Extensions) 5-23
ECHO 5-23
FOR 5-24
GOTO 5-25
IF 5-26
PAUSE 5-27
REM 5-27
SHIFT 5-27

MS-DOS Editing and Function Keys 5-28
Special MS-DOS editing keys 5-29
Control character functions 5-32

MS-DOS Commands

The following MS-DOS commands are described in this chapter. Synonyms for commands are enclosed in parentheses.

BREAK	Sets CONTROL-C check
CD	Changes directories and prints working directory (CHDIR)
CHKDSK	Scans the directory of the default or designated drive and checks for consistency
CLS	Clears screen
COPY	Copies file(s) specified
CTTY	Changes console TTY
DATE	Displays date and allows changes
DEL	Deletes file(s) specified (ERASE)
DIR	Lists requested directory entries
DISKCOPY	Copies disks
EXE2BIN	Converts executable files to binary format
EXIT	Exits command and returns to lower level
FIND	Searches for a specified text string
FORMAT	Formats a disk in MS-DOS format
MD	Makes a directory (MKDIR)
MORE	Displays the output one screen at a time
PATH	Specifies a path for use by MS-DOS when searching for commands
PRINT	Background print feature
PROMPT	Designates command prompt
RD	Removes specified directory (RMDIR)
RECOVER	Recovers a bad disk
REN	Renames the first file as the second file (RENAME)
SET	Sets one string value to another
SORT	Sorts data alphabetically, in ascending or descending order
SYS	Transfers MS-DOS system files from drive A: to the drive specified
TIME	Displays time and allows changes
TYPE	Displays the contents of the specified ASCII file
VER	Displays MS-DOS version number
VERIFY	Verifies writes to disk
VOL	Displays the volume identification number

Batch Commands (Command Extensions)

ECHO	Turns the batch file echo feature on/off
FOR	Batch command extension
GOTO	Batch command extension
IF	Batch command extension
PAUSE	Pauses for input in a batch file
REM	Displays a comment in a batch file
SHIFT	Increases the number of replaceable parameters in batch process

For instructions on the EDLIN, EXE2BIN, and LINK utilities, consult the Sanyo System Software MS-DOS Reference Manual, available at your authorized Sanyo computer dealer.

BREAK Internal command

Format: BREAK [ON | OFF]

Purpose: Sets the Ctrl-C check.

If you are running an application program that uses Ctrl-C as a function key, you should deactivate the MS-DOS Ctrl-C function so that when you press Ctrl-C you will return to your program and not the operating system. Specify BREAK OFF to deactivate Ctrl-C and BREAK ON when you have finished running your application program and are using MS-DOS.

CD (CHANGE DIRECTORY)

Internal command

Format: CD [*pathname*]

Synonym CHDIR

Purpose: Changes directory to a different path and displays current (working) directory.

If you want to change your path to another directory (such as \BIN\USER\JOE), type:

```
CD  \BIN\USER\JOE
```

A shorthand notation is also available with this command:

```
CD ..
```

This command will always put you in the parent directory of your working directory.

CD used without a pathname displays your working directory.

CHKDSK (CHECK DISK) External command

Format: CHKDSK [d:][*filespec*][/F][/V]

Purpose: Scans the directory of the specified disk drive and checks it for consistency.

Run CHKDSK occasionally on each disk to check for errors in the directory. If any errors are found, CHKDSK will display error messages, if any, and then a status report.

A sample status report follows:

```
160256  bytes total disk space
  8192  bytes in 2 hidden files
   512  bytes in 2 directories
 30720  bytes in 8 user files
121344  bytes available on disk

 65536  bytes total memory
 53152  bytes free
```

CHKDSK will not correct the errors found in your directory unless you specify the /F (fix) switch. Typing /V causes CHKDSK to display messages while it is running.

CLS Internal command

Format: CLS

Purpose: Clears the screen.

The CLS command causes MS-DOS to send the ANSI escape sequence ESC[2J (which clears your screen) to display.

COPY Internal command

Format: COPY *filespec* [*filespec* | *pathname*][/V]

Purpose: Copies one or more files to another disk. If you prefer, you can copy the file to the same disk. Used this way, the COPY command can be used to copy files onto the same disk.

If the second parameter is not specified, the copy will be on the default drive and will have the same name as the original file. While this is possible when the default drive is different from the drive that contains the original program, copying files to themselves is not allowed.

The second parameter may take three forms:

1. If the second parameter is a drive designation (*d* :) only, the original file is copied using the original filename to the designated drive.

2. If the second parameter is a filename only, the original file is copied to a file on the default drive with the filename specified.

3. If the second parameter is a full filespec, the original file is copied to a file on the default drive with the filename specified.

The /V switch causes MS-DOS to verify that the sectors written on the destination disk are recorded properly. Although recording errors when running COPY are rare, you can use this parameter to verify that your data has been correctly recorded. The COPY command also allows file concatenation (appending) while copying. Concatenation is accomplished by simply listing any number of files as parameters to COPY, separated by +

For example,

```
COPY  A.XYZ+B.COM+B:C.TXT  BIGFILE.CRP
```

This command concatenates files named A.XYZ, B.COM, and B:C.TXT and places them in the file on the default drive called BIGFILE.CRP.

CTTY Internal command

Format: CTTY *device*

Purpose: Allows you to change the device from which you issue commands (TTY represents the console).

The *device* is the device from which you are giving commands to MS-DOS. This command is useful if you want to change the device on which you are working. The following command moves all command I/O (input/output) from the current device (the console) to the AUX port, such as an RS-232C port.

```
CTTY  AUX
```

DATE Internal command

Format: DATE [*mm – dd – yy*]

Purpose: Enters or changes the system date. This date will be recorded in the directory for any files you create or alter.

If you type DATE, DATE will respond with the message:

```
Current date is mm – dd – yy
Enter new date:
```

Press ENTER if you do not want to change the date shown. You can also type a date after the DATE command, as in:

```
DATE  7–25–84
```

DEL (DELETE) Internal command

Format: DEL [*filespec* | *pathname*]

Synonym: ERASE

Purpose: Deletes all files with the designated filespec.

If the filespec is *.*, the prompt, "Are you sure?" appears. If a Y or y is typed as a response, then all files are deleted as requested. You can also type ERASE for the DELETE command.

DIR (DIRECTORY) Internal command

Format: DIR [*filespec* | *pathname*][/P][/W]

Purpose: Lists the files in a directory.

If you type DIR, all directory entries on the default drive are listed. If only the drive specification is given (DIR d:), all entries on the disk in the specified drive are listed.

If you type only a filename with no extension (DIR filename), all files having the filename embedded in their names on the disk in the default drive are listed. In all cases, files are listed with their size in bytes and with the time and date of their last modification.

The wild card characters ? and * (question mark and asterisk) may be used in the filename parameter. The following DIR commands are equivalent:

COMMAND	EQUIVALENT
DIR	DIR *.*
DIR FILENAME	DIR FILENAME.*
DIR .EXT	DIR *.EXT

The /P switch selects the Page Mode. With /P, display of the directory pauses after the screen is filled. To resume display of output, press any key.

The /W switch selects Wide Display. With /W, only filenames are displayed.

DISKCOPY External command

Format: DISKCOPY [d:] [d:]

Purpose: Copies the disk in the source drive to the disk in the destination drive.

The first parameter you specify is the source drive. The second parameter is the destination drive.

If the drives designated are the same, disk copy is done using a single drive. In this process, MS-DOS will give you prompts to insert and remove the disks at the appropriate times. DISKCOPY waits for you to press any key before continuing.

Notes:

1. If you omit both parameters, a single-drive copy operation will be performed on the default drive.

2. If you omit the second parameter, the default drive will be used as the destination drive.

EXIT Internal command

Format: EXIT

Purpose: Exits the program COMMAND.COM (the command processor) and returns to a previous level, if one exists.

This command can be used when you are running an application program and want to start the MS-DOS command processor as a child process and later return to your program.

FC File comparison utility

Format: FC [/#][/B][/W][/C] *filename1 filename2*

Purpose: Compares the two files and display the differences.

The comparisons are made in one of two ways: on a line-by-line or a byte-by-byte basis. The line-by-line comparison isolates blocks of lines that are different between the two files and prints those blocks of lines. The byte-by-byte comparison displays the bytes that are different between the two files.

There are four switches that you can use with FC.

/B Specifies a binary comparison. The two files are compared on a byte-to-byte basis. The mismatches are printed as follows:

```
—ADDRS——F1——F2—
 xxxxxxxx   yy   zz
```

(where xxxxxxxx is the relative address of the pair of bytes from the beginning of the file).

/# # stands for a number from 1 to 9. This switch specifies the number of lines that must match after a difference has been found for the files to be evaluated as matching.

/W Causes FC to compress whites (tabs and spaces) during comparison.

/C Causes the matching process to ignore the differences between upper- and lowercase letters.

FC reports the differences between the two files you specify by displaying the first filename, followed by the lines that differ between the files, followed by the first line to match in both files. FC then displays the second filename, followed by the lines that are different, followed by the first line that matches. The default for the number of lines to match between the files is 3. For example,

```
...
...

----------<filename1>
<difference>
<1st line to match file2 in file1>

----------<filename2>
<difference>
<1st line to match file1 in file2>

-------------------------------

...
...
```

FC will continue to list the difference.

If there are too many differences (involving too many lines), the program will simply report that the files are different and stop.

If no matches are found after the first difference is found, FC will display:

*** Files are different ***

FIND External command

Format: FIND [/V][/C][/N] *string* [*filename...*]

Purpose: Searches for a specific string of text in a file or files.

FIND is a filter that takes as parameters a string and a series of filenames. If no files are specified, FIND will take the input on the screen and display all lines that contain the specified string.

Switches for FIND are:

/V display all lines not containing the specified string.

/C print only the count of lines that contained a match in each of the files.

/N causes each line to be preceded by its relative line number in the file.

The string should be enclosed in quotes. For example, the following command displays all lines from BOOK1.TXT and BOOK2.TXT (in that order) that contain the string "Fool's Paradise."

```
FIND "Fool's Paradise" BOOK1.TXT BOOK2.TXT
```

The following command causes MS-DOS to display all names of the files on the disk in drive B: which do not contain the string DAT.

```
DIR B: | FIND /V "DAT"
```

FORMAT External command

Format: FORMAT [*d*:][/1][/8][/V][/S]

Purpose: Formats the disk in the specified drive to accept MS-DOS files.

This command initializes the directory and file allocation tables. If no drive is specified, the disk in the default drive is formatted.

The /1 switch causes FORMAT to produce a single sided disk.

The /8 switch causes FORMAT to produce a disk whitch is eight sector. If not specify, produce a nine sector disk.

The /V switch causes FORMAT to prompt for a volume label after the disk is formatted.

If the /S switch is specified, it must be the last switch typed; FORMAT then copies the operating system files from the disk in the default drive to the newly formatted disk.

MD Internal command

Format: MD *pathname*

Synonym: MKDIR

Purpose: Makes a new directory.

The following command will create a subdirectory \USER in your root directory.

```
MKDIR  \USER
```

To create a directory named JOE under \USER, type:

```
MKDIR  \USER\JOE
```

MORE External command

Format: MORE

Purpose: Sends output to console one screen at a time.

MORE is a filter that takes a standard input and displays one screen of information at a time. The MORE command then pauses and displays the —MORE— message at the bottom of your screen.

Press the ENTER key to display the next screen of information. The process continues until all the input data has been read.

The MORE command is useful for viewing a long list or file. If you type the following, MS-DOS will display the file MYFILES.COM one screen at a time.

```
TYPE  MYFILES.COM \ MORE
```

PATH Internal command

Format: PATH [*pathname* [; *pathname*]...]

Purpose: Sets a command path.

This command gives MS-DOS the names of directories to search for external commands.

To tell MS-DOS to search the \BIN\USER\JOE directory for external commands, type:

```
PATH  \BIN\USER\JOE
```

MS-DOS will now search the \BIN\USER\JOE directory for external commands until you set another path.

The command PATH with no parameters will print the current path.

PRINT External command

Format: PRINT [[*filespec*][/T][/C][/S]]...

Purpose: Prints a text file on a line printer while processing other MS-DOS commands (this is usually called "background printing").

The following switches are provided with this command:

/T TERMINATE: This switch removes all files waiting to be printed in the print queue.

/C CANCEL: This switch turns on the cancel mode.

/P PRINT: This switch turns on the print mode.

PRINT with no parameters displays the contents of the print queue on your screen without changing the queue.

PROMPT Internal command

Format: PROMPT [*prompt-text*]

Purpose: Changes the MS-DOS command prompt.

The following characters can be used in the prompt command to specify special prompts. They must all be preceded by a dollar sign ($) in the prompt command:

Character	Prompt
$	– The '$' character
t	– The current time
d	– The current date
p	– The current directory of the default drive
v	– The version number
n	– The default drive
g	– The '>' character
l	– The '<' character
b	– The '|' character
_	– A CR LF sequence
s	– A space (leading only)
h	– A backspace
e	– ASCII code X'1B' (escape)

If no text is typed, the prompt will be set to the default prompt.

For example, the following command specifies a two-line prompt which gives the time and date.

```
PROMPT  Time = $t$_Date = $d
```

Resulting prompt:

```
Time = (current time)
Date = (current date)
```

RD (REMOVE DIRECTORY)

Internal command

Format: RD *pathname*

Synonym: RMDIR

Purpose: Removes a directory from a hierarchical directory structure.

The directory to be removed must be empty except for the . and .. shorthand symbols.

To remove the \BIN\USER\JOE directory, first use the DIR command for that path to ensure that the directory does not contain any important files that you do not want deleted. Then type:

```
RMDIR  \BIN\USER\JOE
```

RECOVER External command

Format: RECOVER { *filename* | *d:* }

Purpose: Recovers a file or an entire disk containing bad sectors.

If a sector on a disk is bad, you can recover either the file containing that sector (excepting the bad sector) or the entire disk (if the bad sector was in the directory).

To recover a specific file, type:

```
RECOVER filename
```

To recover a disk, type:

```
RECOVER d:
```

where *d:* is the letter of the drive containing the disk to be recovered.

REN (RENAME) Internal command

Format: REN *filespec filename*

Synonym: RENAME

Purpose: Changes the name of the first parameter (filespec) to the second parameter (filename).

The wild card characters may be used in either parameter. All files matching the first filespec are renamed.

If wild card characters appear in the second filename, the corresponding character positions will not be changed. For example, the following command changes the names of all files with the .LST extension to similar names with the .PRN extension:

```
REN *.LST *.PRN
```

SET Internal command

Format: SET [*string=string*]

Purpose: Specifies strings to be equivalent for use in later programs.

This command is meaningful only if you want to specify strings that will be used by programs you have written.

The SET command can also be used in batch processing. If your batch file contains the statement "LINK %FILE%", you can set the name that MS-DOS will use for that variable with the SET command. The command SET FILE = DOMORE replaces the %FILE% parameter with the filename DOMORE.

Note that when you use text (instead of numbers) as replaceable parameters, the name must be ended by a percent sign.

SORT External command

Format: SORT [/R][/+*n*]

Purpose: SORT reads from standard input, sorts the data, then writes it to standerd output.

There are two switches which allow you to select parameters:

/R sort from Z to A.

/+n sort starting with column n where n is number.

Examples:

The following command will pipe the output of the directory command to the SORT filter. The SORT filter will sort the directory listing starting with column 14 (the column that contains the file size), then send the output to the console. Thus, the result of this command is a directory sorted by file size:

DIR ¦ SORT /+14

SYS (SYSTEM) External command

Format: SYS *d:*

Purpose: Transfers the MS-DOS system program from the disk in the default drive to the disk in the drive specified by *d:* .

SYS is normally used to place the system on a formatted disk which contains no files. An entry for *d:* is required.

IO.SYS and MSDOS.SYS are both copied. These are hidden files that do not appear when the DIR command is executed.

Note that COMMAND.COM (the command processor) is not transferred. Use the COPY command to transfer COMMAND.COM.

TIME Internal command

Format: TIME [*hh* [: *mm*]]

Purpose: Enters or changes the system time.

If the TIME command is entered without entering parameters, the following message is displayed:

Current time is *hh* : *mm* : *ss* . *cc*
Enter new time:

Press the ENTER key if you do not want to change the time. A new time may be given as an parameter to the TIME command as in:

TIME 8:20

MS-DOS uses the time entered as the new time if the parameters and delimiters are valid.

TYPE Internal command

Format: TYPE *filespec*

Purpose: Displays the contents of the file on the console screen.

The only formatting performed by TYPE is that tabs are expanded to spaces consistent with tab stops every eighth column. Note that if you display binary files with the TYPE command, control characters (such as Ctrl-Z) are sent to your computer, including bells, form feeds, and escape sequences.

VER Internal command

Format: VER

Purpose: Prints MS-DOS version number.

If you want to know what version of MS-DOS you are using, type VER. The version number will be displayed on your screen.

VERIFY Internal command

Format: VERIFY [ON | OFF]

Purpose: Turns the verify switch on or off when writing to disk.

This command has the same purpose as the /V switch in the COPY command.

VERIFY ON remains in effect until you issue a VERIFY OFF command to MS-DOS.

If you want to know what the current setting of VERIFY is, type VERIFY with no parameters.

VOL (VOLUME) Internal command

Format: VOL [*d:*]

Purpose: Displays the disk volume label.

This command prints the volume label of the disk in drive *d:*. If no drive is specified, MS-DOS prints the volume label of the disk in the default drive.

Batch Commands (Command Extensions)

The following commands are batch processing commands. They add flexibility and power to your batch programs. The commands discussed below are ECHO, FOR, GOTO, IF, PAUSE, REM, and SHIFT.

If you will not be writing batch programs, you may skip this section.

ECHO

Format: ECHO [ON | OFF | *message*]

Purpose: Turns the batch echo feature on and off.

Normally, commands in a batch file are displayed ("echoed") on the console. ECHO OFF turns off this feature. ECHO ON turns the echo back on.

If ON or OFF are not specified, the current setting is displayed.

If the *message* is specified, the message is displayed on the screen.

FOR

Format: FOR %%*c* IN *set* DO *command*
– for batch processing

FOR %*c* IN *set* DO *command*
– for interactive processing

Purpose: Command extension used in batch and interactive file processing.

c can be any character except 0,1,2,3,..,9. This avoids confusion with %0-%9 used as batch parameters.

set is (*item item* ...)

The %%*c* variable is set sequentially to each member of *set* , and *command* is evaluated.

Examples:

FOR %%f IN (*.PRN) DO PRINT %%f

FOR %%f IN (FOO BAR BLECH) DO REM %%f

The "%%" is needed so that after batch parameter (%0-%9) processing is done, there is one "%" left. If FOR is not in a batch file, only one "%" should be used.

GOTO

Format: GOTO *label*

Purpose: Command extension used in batch file processing.

GOTO causes commands to be taken from the batch file beginning with the line after the *label* definition. If no label has been defined, the current batch file will terminate.

Example:

```
:foo
REM looping...
GOTO foo
```

will produce an infinite sequence of messages: REM looping....

Starting a line in a batch file with ":" causes the line to be ignored by batch processing.

The characters following GOTO define a label; this procedure may also be used to input comment lines.

IF

Format: IF *condition command*

Purpose: Command extension used in batch file processing.

The parameter *condition* is one of the following:

ERRORLEVEL *number*
True if the previous program executed by COMMAND had an exit code of *number* or higher.

string1 = *string2*
True if *string1* and *string2* are identical after parameter substitution. Strings may not have embedded delimiters.

EXIST *filename*
True if *filename* exists.

NOT *condition*
True if *condition* is false.

The IF statement allows conditional execution of commands. When the *condition* is true, then the *command* is executed. Otherwise, the *command* is ignored.

Examples:

```
IF  NOT  EXIST  \TMP\FOO  ECHO  Can't find file

IF  NOT  ERRORLEVEL 3  LINK  $1,,;
```

PAUSE

Format: PAUSE [*comment*]

Purpose: Suspends execution of the batch file.

PAUSE suspends execution until you press any key, except Ctrl-C.

When the command processor encounters PAUSE, it prints the following message:

Strike a key when ready ...

Comments are optional and may be entered on the same line as PAUSE. An optional prompt message may be given in such cases. The prompt message is displayed before the "Strike a key" message.

REM (REMARK)

Format: REM [*comment*]

Purpose: Displays remarks during execution of a batch file.

The only delimiters allowed in comments are spaces, tabs, and commas.

SHIFT

Format: SHIFT

Purpose: Allows more than 10 replaceable parameters to be used in batch file processing.

Command files are usually limited to 10 parameters, %0 through %9.

If there are more than 10 parameters in a command line, those that appear after the 10th (%9) will be shifted one at a time into %9 by successive shifts.

MS-DOS Editing and Function Keys

Special MS-DOS editing keys

Because MS-DOS saves the last command line in a template, you do not have to type the same key sequence twice.

Use the template and the special editing keys to take advantage of the following features:

1. A command line can be instantly repeated by pressing two keys.

2. If you make a mistake in the command line, you can edit it and reenter it without having to retype the entire command line.

3. A command line that is similar to a preceding command line can be edited and executed with a minimum of typing by pressing a special editing key.

When you enter a command to MS-DOS on a command line and press ENTER key, the command is automatically sent to the command processor (COMMAND.COM) for execution. When this is done, the command is copied on the template. You can now recall the command or modify it with MS-DOS special editing keys.

Key	Editing function
F1 or →	Copies one character from the template to the command line
F2	Copies characters up to the character specified in the template and puts these characters on the command line
F3	Copies all remaining characters in the template to the command line
DEL	Skips over (does not copy) a character in the template
F4	Skips over (does not copy) the characters in the template up to the character specified
ESC	Voids the current input; leaves the template unchanged
INS	Enters/exits insert mode
F5	Enters the current line as the template
F6	Puts a Ctrl-Z (1AH, the end-of-file character) in the new template

Examples:

If you type the following command, MS-DOS displays information about the file PROG.COM on your screen.

```
DIR PROG.COM
```

The command line is also saved in the template. To repeat the command, just press F3 and ENTER.

The command is redisplayed on the screen as you type, as shown below:

```
<F3>DIR PROG.COM <ENTER>
```

If you want to display information about a file named PROG.ASM, you can use the contents of the template and type:

```
<F2>C
```

Typing F2 C copies all characters from the template to the command line, up to but not including C. MS-DOS displays:

```
DIR PROG.
```

In this example, the underline is your cursor. Next type:

```
.ASM
```

The result is:

```
DIR PROG.ASM
```

The command line DIR PROG.ASM is now in the template and ready to be sent to the command processor for execution. To do this, press ENTER.

Now assume that you want to execute the following command:

```
TYPE PROG.ASM
```

To do this, type:

```
TYPE<Ins>  <F3><ENTER>
```

The characters "TYPE" replace "DIR " in the template. To insert a space between "TYPE" and "PROG.ASM", press Ins and the space bar. Finally, to copy the rest of the template to the command line, press F3 and ENTER. The command TYPE PROG.ASM has been processed by MS-DOS.

If you had misspelled TYPE as BYTE, you could save the misspelled line before you press ENTER by creating a new template with the F5 key:

BYTE PROG.ASM<F5>

You could then edit this erroneous command by typing:

T<F1>P<F3>

The F1 key copies a single character from the template to the command line. The resulting command line is then the command that you want:

TYPE PROG.ASM

Control character functions

Control character functions are functions that affect the command line. You have already learned about CONTROL-BREAK. Other similar control character functions are described below.

Remember that when you type a control character, such as Ctrl-C, you must hold down the Ctrl key and then press the "C" key.

Control character functions

Control character	Function
Ctrl-C	Aborts current command. Equivalent to CONTROL-BREAK.
Ctrl-H	Removes last character from command line and erases character from screen. Equivalent to BACKSPACE (←).
Ctrl-J	Inserts physical end-of-line, but does not empty the command line. Use Ctrl-J key to extend the current logical line beyond the physical limits of one terminal screen.
Ctrl-P	Toggles output from screen to line printer. Equivalent to HARDCOPY.
Ctrl-S	Suspends output display on screen. Press any key to resume. Equivalent to STOP SYSTEM.
Ctrl-X	Cancels the current line; empties the command line; then outputs a back slash (\), carriage return, and line feed. The template used by the special editing commands is not affected.

CHAPTER 6.
TECHNICAL REFERENCE

Specifications 6-2
Character Codes 6-4
Basic Reserved Words 6-7
Basic Error Messages 6-9
Disk Error Messages 6-12
Memory Map 6-15
Interrupt Vectors 6-16
    Bios interrupts 6-16
    Dos interrupts 6-17
Interrupt Routines 6-18
    Interrupt 10H 6-18
    Interrupt 11H 6-20
    Interrupt 12H 6-21
    Interrupt 13H 6-21
    Interrupt 16H 6-22
    Interrupt 17H 6-22
I/O Map 6-24
DIP Switches 6-25
Expansion Slot Signals 6-26
Connector Assignment 6-27
    Keyboard 6-27
    RGB moniter 6-27
    Composite video 6-27
    Parallel centronics-type printer 6-28
DOS Editing Keys 6-29

Specifications

Hardware

CPU:	8088 (8.00 MHz)
ROM:	16K (IPL/CG)
RAM:	256K
Speaker:	Errors and sounds
VIDEO-RAM:	16K
Graphics:	640 × 200 dots (Monochrome) 320 × 200 dots (Color)

Floppy disk drive:

MBC-770 360K × 1
MBC-775 360K × 2

Printer interface: Parallel Centronics-type

Dimensions / Weight

Main unit:	526 (width) × 224 (height) × 420 (depth) 20 3/4" (w) × 8 4/5" (h) × 16 1/2" (d)
	17.5 kg / 38.9 lbs.
Keyboard:	500 (width) × 363 (height) × 190 (depth) 19 3/4" (w) × 14 1/2" (h) × 7 1/2" (d)
	1.9 kg / 4.2 lbs.

Electrical

Voltage:	Local voltage ±10 %
Frequency:	50/60 Hz
Power consumption:	80 W

Environmental

Ambient temperature

Operating:	50 to 95°F (10 to 35°C)
Quiescent:	5 to 149°F (–15 to 65°C)

Relative humidity

Operating:	30% to 80%, no condensation
Quiescent:	20% to 80%, no condensation

Character Codes

Explanation

The character codes used in the Sanyo MBC-770 series computer are listed in the chart below. There are 256 possible codes. The character set conbines both standard 7-bit ASCII charactors and graphic characters (those on the second half of the set that have bit 8 high). The numbers running along the top row represent the four high order bits (nibble).

B7 B6 B5 B4

The numbers running down the side column represent the low order four bits (nibble).

B3 B2 B1 B0

The combination of these two "nibbles" represent the character code byte.

Character set (00–7F)

DECIMAL VALUE ➡ / HEXA DECIMAL VALUE ⬇	0	16	32	48	64	80	96	112
	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	'	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	-	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

Character set (80–FF)

DECIMAL VALUE ➡		128	144	160	176	192	208	224	240
⬇	HEXA DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	1/4 Dots On			∝	≡
1	1	ü	æ	í	1/2 Dots On			β	±
2	2	é	Æ	ó	3/4 Dots On			γ	≧
3	3	â	ô	ú				π	≦
4	4	ä	ö	ñ				Σ	⌠
5	5	à	ò	Ñ				σ	⌡
6	6	å	û	ª				μ	÷
7	7	ç	ù	º				τ	≈
8	8	ê	ÿ	¿				Φ	°
9	9	ë	Ö	⌐				⊖	●
10	A	è	Ü	¬				Ω	•
11	B	ï	¢	1/2				δ	√
12	C	î	£	1/4				∞	η
13	D	ì	¥	¡				∅	²
14	E	Ä	Pts	«				∈	▌
15	F	Å	f	»				∩	BLANK 'FF'

Basic Reserved Words

The following words are reserved words in Sanyo GW-BASIC 2.0. They cannot be used as variable names unless they are embedded.

ABS
ALL
AND
AS
ASC
ATN
AUTO

BASE
BEEP
BLOAD
BSAVE

CALL
CDBL
CHAIN
CHDIR
CHR$
CINT
CIRCLE
CLEAR
CLOSE
CLS
COLOR
COM
COMMON
CONT
COS
CSNG
CSRLIN
CVD
CVI
CVS

DATA
DATE$
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DRAW

EDIT
ELSE
END
ENVIRON
ENVIRON$
EOF
EQV
ERASE
ERDEV
ERDEV$
ERL
ERR
ERROR
EXP

FIELD
FILES
FIX
FN
FOR
FRE

GET
GOSUB
GOTO

HEX$

IF
IMP
INKEY$
INP
INPUT
INPUT#
INPUT$
INSTR
INT

KEY
KILL

LEFT$
LEN
LET
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LPOS
LPRINT
LSET

MERGE
MID$
MKD$
MKDIR
MKI$
MKS$
MOD

NAME
NEW
NEXT
NOT

OCT$
OFF
ON
OPEN
OPTION
OR
OUT

PAINT
PEEK
PEN
PLAY
PMAP
POINT
POKE
POS
PRESET
PRINT
PRINT#
PSET
PUT

RANDOMIZE
READ
REM
RENUM
RESET
RESTORE
RESUME
RETURN
RIGHT$
RMDIR
RND
RSET
RUN

SAVE
SCREEN
SEG
SGN
SHELL
SIN
SOUND
SPACE$
SPC(
SQR
STEP
STICK
STOP
STR$
STRIG
STRING$
SUB
SWAP
SYSTEM

TAB(
TAN
THEN
TIME$
TO
TROFF
TRON

USING
USR

VAL
VARPTR
VARPTR$
VIEW

WAIT
WEND
WHILE
WIDTH
WINDOW
WRITE
WRITE#

XOR

Basic Error Messages

Number	Message
1	**NEXT without FOR** A FOR statement did not precede the NEXT statement.
2	**Syntax error** Syntax error (misspeling)
3	**RETURN without GOSUB** A GOSUB statement did not precede the RETURN statement.
4	**Out of DATA** The number of constants in the DATA statement was less than the number of variables required by the READ statement.
5	**Illegal function call** An Illegal statement or function call was found.
6	**Overflow** The result of calculation was too large.
7	**Out of memory** There is more room in memory to execute the program or dimension the array.
8	**Undefined line number** The specified line number was not found.
9	**Subscript out of range** An Array subscript was larger than that specified in DIM statement or was larger than 10 when no DIM statement was used.
10	**Duplicate definition** There are two DIM statements for the same variable or a definition was duplicated.
11	**Division by zero** A number was divided by 0.

12 **Illegal direct**
The statement input cannot be used in the direct mode.

13 **Type mismatch**
The variable type did not match the data type.

14 **Out of string space**
The area reserved for character strings in memory was exhausted.

15 **String too long**
The charactor string waslonger than 255 characters.

16 **String Formula too complex**
The string expression was too long or too complex.

17 **Can't continue**
The program cannot be continued.

18 **Undefined user function**
The function was not defined using DEF FN.

19 **No RESUME**
A RESUME statement was not found in the error processing routine.

20 **RESUME without error**
A RESUME statement was executed although no error has occured.

21 **Unprintable error**
Undefined.

22 **Missing operand**
An expression contained an operator without an operand.

23 **Line buffer overflow**
The line entered was too long.

24 **Device timeout**
The specified device is not available.

25 **Device fault**
An incorrect device has been specified.

26 **FOR without NEXT**
An NEXT statement was not found in the FOR/NEXT loop.

27 **Out of paper**
The printer is out of paper.

28 **Unprintable error**
Undefined.

29 **WHILE without WEND**
A WEND statement was not found in the WHILE/WEND loop.

30 **WEND without WHILE**
A WHILE statement was not found before a WEND was encountered.

31-49 **Unprintable error**
Undefined.

Disk Error Messages

Number	Message
50	**Field overflow** The length defined in the FIELD statement exceeded the specified record length; or the string has more than 255 characters.
51	**Internal error** An internal malfunction occurred.
52	**Bad file number** The file number is invalid.
53	**File not found** The specified file was not found on the disk.
54	**Bad file mode** The file open mode was incorrect.
55	**File already open** The specified file was already opened; or a KILL was attempted on a file still open.
56	**Unprintable error** Undefined.
57	**Device I/O error** An error occurred during a disk I/O operation.
58	**File already exists** The specified file already exists.
59-60	**Unprintable error** Undefined.
61	**Disk full** The disk is full; no more data can be saved on it.
62	**Input past end** The read instruction went past the end of file.

63 **Bad record number**
The record number is invalid.

64 **Bad file name**
The file name is invalid.

65 **Unprintable error**
Undefined.

66 **Direct statement**
A statement was found that did not have in file a line number during LOADing of the file.

67 **Too many files**
Too many files were opened.

68 **Device Unavailable**
The specified device is not available at this time.

69 **Communications buffer overflow**
The communications buffer is out of space.

70 **Disk write protected**
The disk has a write-protect seal on it.

71 **Disk not ready**
The disk drive is not ready to operate.

72 **Disk media error**
The disk is probably damaged.

74 **Rename across disks**
The name was incorrectly specified.

75 **Path/file access error**
A correct Path could not be found for the file name.

76 **Path not Found**
The specified path could not be found.

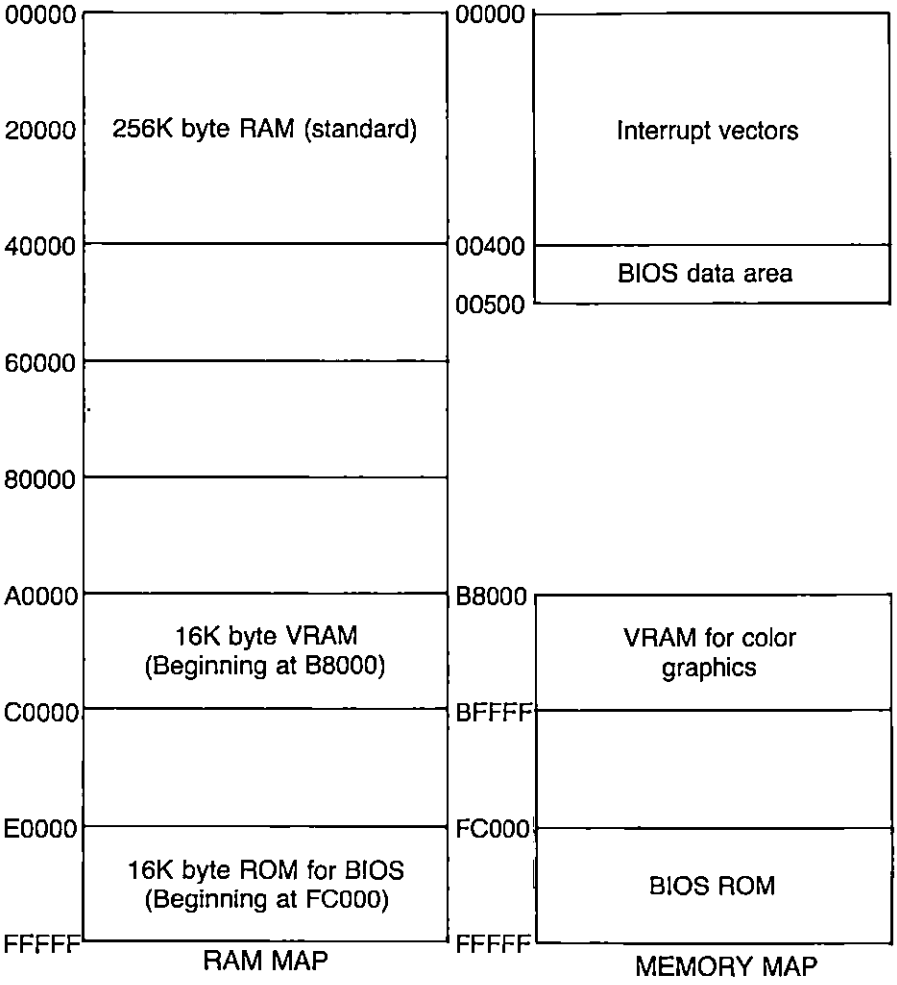
** **You cannot SHELL to BASIC**
BASIC was called from BASIC.

** **Can't continue after SHELL**
There is not enough memory to continue BASIC after executing the SHELL statement.

Memory Map

(Numbers in Hexadecimal)

Starting address



Interrupt Vectors

BIOS interrupts

Location in memory	Interrupt number	Function
0 – 3	0	Division by zero
4 – 7	1	Single step
8 – B	2	Non-maskable interrupt
C – F	3	Break point
10 – 13	4	Overflow
14 – 17	5	Print screen
18 – 1B	6	Reserved
1C – 1F	7	Reserved
20 – 23	8	Timer
24 – 27	9	Keyboard
28 – 2B	A	Reserved
2C – 2F	B	Reserved
30 – 33	C	Reserved
34 – 37	D	Reserved
38 – 3B	E	Disk drive
3C – 3F	F	Printer
40 – 43	10	Video I/O
44 – 47	11	Device check
48 – 4B	12	Memory check
4C – 4F	13	Disk I/O
50 – 53	14	RS–232C I/O
54 – 57	15	Reserved
58 – 5B	16	Keyboard I/O
5C – 5F	17	Printer I/O
60 – 63	18	BASIC entry
64 – 67	19	Bootstrap
68 – 6B	1A	Time of day
6C – 6F	1B	Keyboard break
70 – 73	1C	Timer
74 – 77	1D	Video initialize
78 – 7B	1E	Disk parameter table
7C – 7F	1F	Character table

DOS interrupts

Location in memory	Interrupt number	Function
80 – 83	20	Terminate program
84 – 87	21	Function request
88 – 8B	22	Terminate address
8C – 8F	23	CTRL-C exit address
90 – 93	24	Fatal error address
94 – 97	25	Absolute disk read
98 – 9B	26	Absolute disk write
9C – 9F	27	Terminate program and maintain it in memory
A0 – FF	28 – 3F	Reserved
100 – 1FF	40 – 7F	Not used
200 – 3C3	80 – F0	Reserved by BASIC
3C4 – 3FF	F1 – FF	Not used

Interrupt Routines

Interrupt 10H

Video I/O

These routines provide the CRT interface.
The following functions are provided:

(AH) = 0 Set mode (AL) contains mode value
Text modes
(AL) = 0 40 × 25 BW
(AL) = 1 40 × 25 color
(AL) = 2 80 × 25 BW
(AL) = 3 80 × 25 color
Graphics modes
(AL) = 4 320 × 200 color
(AL) = 5 320 × 200 BW
(AL) = 6 640 × 200 BW

(AH) = 1 Set cursor type
(CH) – Bits 4-0 = Start line for Cursor
(CL) – Bits 4-0 = End line for Cursor

(AH) = 2 Set cursor position
(DH,DL) – Row,column (0,0) is upper left
(BH) – Page number
(must be 0 for graphics modes)

(AH) = 3 Read cursor position
(BH) – Page number (must be 0 for graphics modes)
On exit:
(DH,DL) – Row,column of current cursor
(CH,CL) – Cursor mode currently set

(AH) = 5 Select active display page (valid only for text modes)
(AL) – New page value
(0-7 for modes 0 and 1, 0-3 for modes 2 and 3)

(AH) = 6 Scroll active page up
(AH) – Number of lines, input lines blanked at bottom of window
AL = 0 means blank entire window
(CH,CL) – Row,column of upper left corner of scroll

(DH,DL) – Row,column of lower right corner of scroll
(BH) – Attribute to be used on blank line

(AH) = 7 Scroll active page down
(AL) – Number of lines, Input lines blanked at top of window
AL = 0 means blank entire window
(CH,CL) – Row,column of upper left corner of scroll
(DH,DL) – Row,column of lower right corner of scroll
(BH) – Attribute to be used on blank line

Character handling routines

(AH) = 8 Read attribute/character at current cursor position
(BH) – Display page (valid for alpha modes only)
On exit:
(AL) – Charcter read
(AH) – Attribute of character read (text modes only)

(AH) = 9 Write attribute/character at current cursor position
(BH) – Display page (valid for text modes only)
(CX) – Count of characters to write
(AL) – Characters to write
(BL) – Attribute of character (alpha)
color of character (graphics)

(AH) = 10 Write character only at current cursor position
(BH) – Display page (valid for alpha modes only)
(CX) – Count of characters to write
(AL) – Character to write

Graphics interface

(AH) = 11 Set color pallette
(BH) – Pallette color ID being set (0–127)
(BL) – Color valueto be used with that color ID

(AH) = 12 Write dot
(DX) – Row number
(CX) – Column number
(AL) – Color value

(AH) = 13 Read dot
(DX) – Row number
(CX) – Column number
(AL) Returns the DOT read

ASCII teletype output routine

(AH) = 14 Write teletype
(AL) – Character to write
(BL) – Foreground color in graphics mode
(BH) – Display page in text mode
NOTE: Screen width is controlled by previous mode set

(AH) = 15 Current video state
Returns the current video state
(AL) – Mode currently set(see AH = 0)
(AH) – Number of character columns on screen
(BH) – Current active display page

CS,SS,DS,ES,BX,CX,DX are preserved by call. All others registers are destroyed.

Interrupt 11H

Equipment determination

This routine returns information what equipment is installed.

Output

(AX) is set, bit significant, to indicate attached I/O

Bit 15,14 – Number of printer attached
Bit 12 – Game I/O attached
Bit 11,10,9 – Number of RS-232C cards attached
Bit 7,6 – Number of disk drive attached

No other registers affected

Interrupt 12H

Memory size determination

This routine returns the amount of memory in the system.

Output

(AX) = Number of contiguous 1K blocks of memory

Interrupt 13H

Disk I/O

This interface provides access to the 5 1/4" disk drives.

Input

(AH) = 0 Reset disk system
(AH) = 1 Read the status of the system into (AL)
Disk status from last operation is used

Registers for READ/WRITE/VERIFY

(DL) – Drive number (0-3 allowed, value checked)
(DH) – Head number (0-1 allowed, not value checked)
(CH) – Track number (0-39, not value checked)
(CL) – Sector number (1-9, not value checked)
(AL) – Number of sectors (max = 9, not value checked)

(ES:BX) – Address of buffer (not required for verify)

(AH) = 2 Read the desired sectors into memory
(AH) = 3 Write the desired sectors from memory
(AH) = 4 Verify the desired sectors

Data variable – Disk pointer (address 00078H – 0007BH)

Double word pointer to the current set of disk parameters.

Output

AH –	Status of operation
CY = 0	Successful operation (AH = 0 on return)
CY = 1	Failed operation (AH has error reason)

For READ/WRITE/VERIFY
DS,BX,DX,CX preserved
AH – Number of sectors acutually read

Interrupt 16H

Keyboard I/O interrupt routine

These routines provide keyboard support.

Input

(AH) = 0	Read the next ASCII character struck from the keyboard Return the result in (AL), scan code in (AH).
(AH) = 1	Set the Z flag to indicate if a code is available to be read. (ZF) = 1 – No code available (ZF) = 0 – Code is available If ZF = 0, the next character in the buffer to be read is in AX, and the entry remains in the buffer.
(AH) = 2	Return the current shift status in AL register.

Interrupt 17H

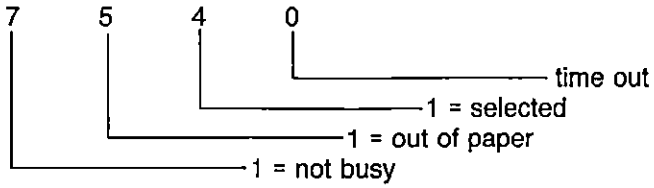
Printer I/O

This routine provides communication with the printer.

(AH) = 0	Print the character in (AL) On return, AH = 1 if chracter could not be printed (time out) Other bits set as normal status call

(AH) = 1 Initialize the printer port returns with (AH) set with printer satatus

(AH) = 2 Read the printer status into (AH)



(DX) -- printer to be used (0,1,2) corresponding to actual values in printer base area

Registers AH is modified
All other unchanged

I/O Map

(Numbers in Hexadecimal)

I/O addresses

Address	Device
00 – 0F	DMA controller 8237
20 – 21	Interrupt controller 8259A
40 – 43	Timer 8253
60 – 63	PPI 8255A
80 – 83	DMA page registers
A0 – AF	NMI mask register
378 – 37F	Centronics printer port
3D0 – 3DF	CRTC 46505
3F0 – 3F7	FDC 765
3F8 – 3FF	Serial port 8250 (option)

DIP Switches

The DIP switches, located at the top center of the main memory board,are shipped with switches 2, 4, 5 and 8 on. This is the setting for most software applications. Please refer to the chart below explaining the functions of the DIP switches.

Switch	Function
1-6	Unused
7-8	Number of 5-1/4 inch disk drives installed

7	8	# of drives
on	on	1
off	on	2
on	off	3
off	off	4

Normal setting is as follows:

Switch #	1	2	3	4	5	6	7	8
Setting		on		on	on			on
	off		off			off	off	

Expansion slot signals

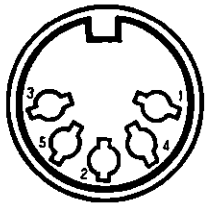
No.	Signal	No.	Signal
B1	Ground	A1	I/O channel check
B2	Reset Drive	A2	Data Bit 7
B3	5V	A3	Data Bit 6
B4	Interrupt Request 2	A4	Data Bit 5
B5	–5V	A5	Data Bit 4
B6	DMA Request 2	A6	Data Bit 3
B7	–12V	A7	Data Bit 2
B8	Reserved	A8	Data Bit 1
B9	12V	A9	Data Bit 0
B10	Ground	A10	I/O Channel Ready
B11	MEMW	A11	Address Enable
B12	MEMR	A12	Address Bit 19
B13	IOW	A13	Address Bit 18
B14	IOR	A14	Address Bit 17
B15	DMA Acknowledge 3	A15	Address Bit 16
B16	DMA Request 3	A16	Address Bit 15
B17	DMA Acknowledge 1	A17	Address Bit 14
B18	DMA Request 1	A18	Address Bit 13
B19	DMA Acknowledge 0	A19	Address Bit 12
B20	Clock	A20	Address Bit 11
B21	Interrupt Request 7	A21	Address Bit 10
B22	Interrupt Request 6	A22	Address Bit 9
B23	Interrupt Request 5	A23	Address Bit 8
B24	Interrupt Request 4	A24	Address Bit 7
B25	Interrupt Request 3	A25	Address Bit 6
B26	DMA Acknowledge 2	A26	Address Bit 5
B27	Timer Count	A27	Address Bit 4
B28	Address Latch Enable	A28	Address Bit 3
B29	5V	A29	Address Bit 2
B30	OSC	A30	Address Bit 1
B31	Ground	A31	Address Bit 0

Connector Assignment

Keyboard

5 Pin DIN connector

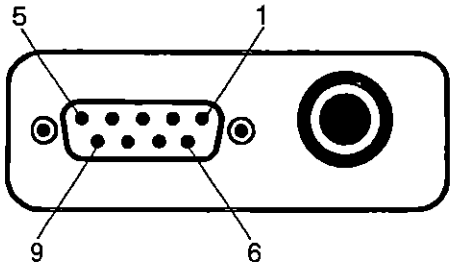
Pin #	Signal
1	Keyboard clock
2	Serial data
3	Reset (from main unit)
4	Ground
5	+5 V



RGB monitor

D-SUB 9 Pin connector

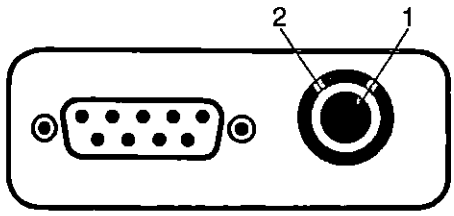
Pin #	Signal
1	Ground
2	Ground
3	Red
4	Green
5	Blue
6	Intensity
7	Reserved
8	Horizontal Sync
9	Vertical Sync



Composite video

Phono jack

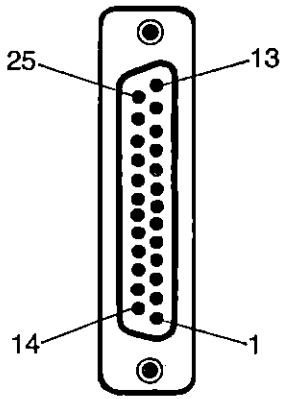
Pin #	Signal
1	Video signal
2	Chassis ground



Parallel centronics-type printer

D-SUB 25 Pin connector

Pin #	Signal
1	Data Strobe
2	Data Bit 0
3	Data Bit 1
4	Data Bit 2
5	Data Bit 3
6	Data Bit 4
7	Data Bit 5
8	Data Bit 6
9	Data Bit 7
10	Acknowledge
11	Busy/Ready
12	Paper Empty
13	Select
14	Auto Feed
15	Error
16	Printer Initalize
17	Select Input
18–25	Ground



DOS Editing Keys

The function keys may be used to make corrections in a command line (in DOS) or when using EDLIN. After a command is entered, it is executed and a record of it is saved in memory. This memory image may act as a "template" for the next command.

F1	Copies and displays one character from the template at a time.
F2	Copies and displsys characters up to the specified character (the next character typed).
F3	Copies the characters remainingn the template to the screen.
F4	Skips over all characters to the specifie dcharacter.
F5	Initializes the template.
F6	Kills the new template (maintains the old template).
INS	Enters/Resets the template insert mode.
DEL	Skips over one character in the template.
←	Backspace and delete.

CHAPTER 7
OPTIONAL PERIPHERAL INSTALLATION

Installation Instructions 7-2
Upper Body Removal 7-3
Top Plate Removal 7-4
Additional Software and Manuals 7-5

Installation Instructions

The instructions contained in this installation guide allow you to install Sanyo peripherals on your MBC-770 series computer. We recommend that you have a qualified technician do the installation. Please consult your authorized Sanyo computer dealer.

Sanyo will assume no responsibility for improper installation due to misinterpretation of the instructions contained in this chapter.

These installation instructions apply only to Sanyo supplied peripherals or their compatible components recommended in this document.

Upper Body Removal

To install Sanyo peripherals, you must remove the upper body and the top plate of the chassis. The instructions below will guide you in doing this. Read the instruction carefully.

Peripheral installations.

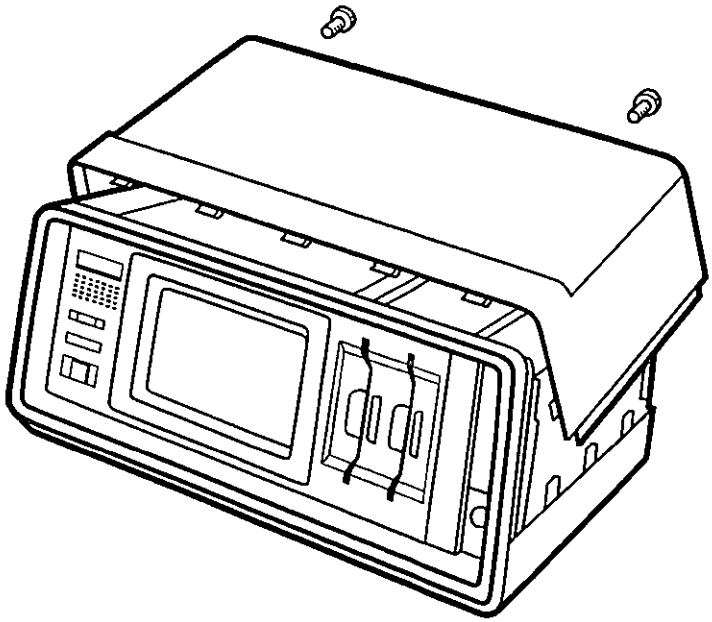
(1) Turn the power off and unplug the AC cord from the electrical outlet.

(2) Unscrew (counter clockwise) the two screws in the connector compartment (one on each side).

(3) Pull the edge of the body up, away from the handle.

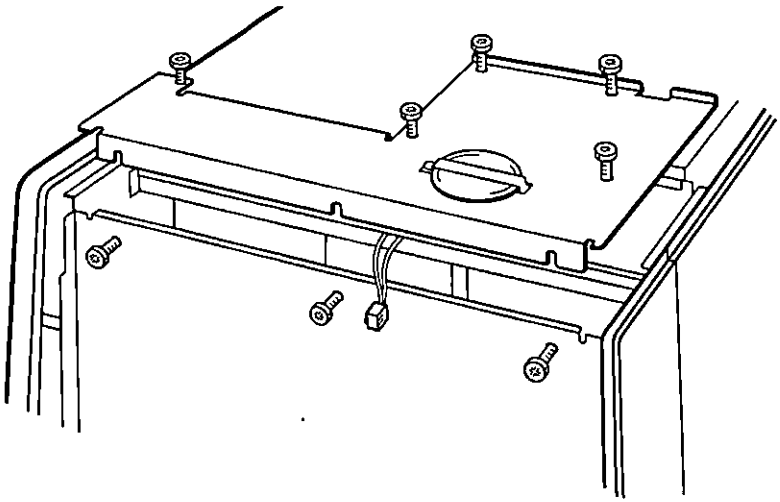
(4) Lift up the upper body.

Refer to the illustration below.



Top Plate Removal

(1) Loosen the screws holding the top panel (3 screws at side, 5 screws at top), and slide the panel toward the front panel. Refer to the illustration below.



(2) Lift up the panel carefully, and disconnect the speaker plug on the main card.

Additional Software and Manuals

Your authorized Sanyo Computer dealer has a wide range of software packages available for your MBC-770 series computer.

In addtion, the following reference manuals are also available from your Sanyo computer dealer.

Sanyo Software Reference Manuals

- Sanyo GW-BASIC Reference Manual
- MS-DOS Reference Manual (Microsoft)
- MS-DOS Macro assembler Manual (Microsoft)
- WordStar Reference Manual (MicroPro)
- SpellStar Reference Manual (MicroPro)
- MailMerge Reference Manual (MicroPro)
- ReportStar Reference Manual (MicroPro)
- DataStar Reference Manual (MicroPro)

CHAPTER 8.
COMPUTER VOCABULARY

Introduction 8-2
Vocabulary 8-2

Introduction

We would like to provide the following glossary so as to clear up possible confusion with the use of the computer terminology in this manual.

Vocabulary

Word	Definition
A>	MS-DOS prompt. The computer is waiting for the next command and the letter indicates the current disk drive, in this case, A is the first disk drive.
ABORT	To end a process by abnormal means.
ACCESS	To read or write data between an external device and the computer.
ADDRESS	A number which addresses a specific memory location. Usually an address is a number expressed in hexadecimal between &H0000 and &HFFFF, or 0 and 65535 in decimal.
ADDRESS BUS	A set of lines to carry the binary encoded address from the CPU to the memory.
ALU	Abbreviation for Arithmetic Logic Unit.
AND	A logic function that is "true" only when all of the input parameters are true.
ANSI	Abbreviation for American National Standards Institute or a standard developed by it.
APPLICATION SOFTWARE	The programs that the operator can use to do a specific task. Examples are a word processor, a spread sheet program, or an inventory program.
ARRAY	A set of data elements organized as a group. One group of variables with subscripts. For example, a one-hundred element array can be defined in BASIC as A(100).

ASCII Acronym for American Standard Code for Information Interchange. ASCII code assigns a unique value from hexadecimal &H00 to &H7E (decimal values from 0 to 127) to the 128 letters, numbers, control characters, and special characters.

ASSEMBLER
A program that converts a source program written in assembly language to machine language. The output of the assembler is called the object code.

ASSEMBLY LANGUAGE
A language that is used when writing machine language programs.

B> MS-DOS prompt. The computer is waiting for the next command. The letter indicates the current disk drive, in this case, the second disk drive.

BACKUP To make a complete copy of the contents of a disk to insure that the data will not be lost.

BASIC Acronym for Beginner's All-purpose Symbolic Instruction Code. BASIC is a high-level language that was first written in the 1960's at Dartmouth College.

BAUD The rate at which serial information flows between two devices.

BINARY A number system based on two different numbers, 0 and 1. Each digit represents a power of two.

BIOS Abbreviation for Basic Input-Output System. This is the section of MS-DOS that controls the hardware of the computer.

BIT A set of the two binary states 0 and 1. A bit is the smallest data unit handled by a computer. Eight bits can be taken together to form larger words called "bytes".

BOOTSTRAP "BOOT"
To initialize the computer when the power is turned on. A program called the bootstrap loader is run immediately at system start; this program reads in the operating system from the floppy disk.

BUFFER A data register or a section of memory that holds data temporarily when doing input and output to speed up the operations. Screen buffers hold information to be sent to the display. Keyboard buffers hold information that was input from the keyboard.

BUS A group of electronic paths (lines) in a computer that allows flow of binary encoded data or addresses between the different parts of the computer.

BYTE A unit of data used to measure the computer's memory size. A byte is made up of eight bits arranged in a row, with bit 7 at the left being the highest significant bit, and bit 0 at the right being the lowest significant bit.

CALL An instruction that calls a subroutine. When CALL is executed, the program execution is transferred to another location.

CHARACTER
A symbol used in the computer. Characters include letters, numbers, graphic symbols, and computer control codes.

CODE A system to represent characters or data for use in the computer. Examples of code systems are ASCII, EBCDIC, and BCD.

COLD START
Starting the computer with the power completely off.

COMPILE To convert a source program written in a high-level language to machine language.

COMPUTER A machine to process instructions. Computer systems can input, process, store, and output data.

CONCATENATE
To append data from one string or data file on to another.

CONNECTOR
A cord with the appropriate lines and plugs used to connect the different parts of the computer.

CONSOLE The keyboard and CRT as a group.

CONSTANT Numerical or alphabetic data that will not change once specified. Pi (3.14159) is a constant.

CONTROL (CTRL) CHARACTER
Characters in ASCII code that are used to control devices such as printers and display screens.

CPU Abbreviation for Central Processing Unit.

CR Abbreviation for the Carriage Return, RETURN, or ENTER key.

CRT Abbreviation for Cathode Ray Tube, which are used in most computers for display.

CURSOR Usually a blinking character that indicates where the next character will be input into the computer.

DATA The term used for information that is processed in a computer.

DB25 A 25-pin connector used with an RS-232C interface.

DECIMAL A numbering system based on ten unique numerals, 0–9.

DEFAULT The value assumed by the application program if the user does not input a value.

DELIMITER A character used to separate two fields during specification of more than one parameter for a function.

DIAGNOSTIC PROGRAM
A program used to evaluate the operational performance of a machine.

DIR The MS-DOS command used to display the directory.

DIRECTORY
The set of filenames on the disk.

DISASSEMBLER
A program that converts machine language to source program.

DISPLAY A video or CRT screen.

DUMP To transfer the contents of memory to the screen or a printer.

EDITOR An application program used to create text files.

ENTRY POINT
The first step of a program to be executed.

EXCLUSIVE OR (EOR)
A binary function that is true only if the inputs are different.

ESCAPE SEQUENCE
A set of control codes, beginning with the code for ESC (&H1E), to control a printer.

EXECUTABLE FILE
A file that has been made into a MS-DOS command.

FAT Abbreviation for File Allocation Table. The FAT is a table written on the disk with a list of the absolute locations of the programs on the disk.

FILE A collection of data on a floppy disk.

FLOPPY A short way to say floppy disk, the 5 1/4-inch or 8-inch media used for data storage in the computer.

FORMAT The method used for specifying data.

FORTRAN Acronym for FORmula TRANslator. This was one of the first compilers available on computers.

FUNCTIONS
A derived sequence of events that manipulate data, variables, and constants to produce a single result.

GRAPHIC CHARACTERS
The character set used for graphic data to be sent to the CRT or the printer.

HARDWARE
The physical components of the computer, as opposed to software, which are programs.

HEXADECIMAL
A numbering system based on sixteen unique numerals (the numbers 0 to 9 and the letters A through F) to represent values. Each digit to the left is a higher power of 16. In BASIC, hexadecimal numbers are preceded by ampersand-H (&H).

HIGH-LEVEL LANGUAGE
A language using English-like commands to control the computer, as opposed to machine language, which is written using numbers only.

HIGH-ORDER BIT
The most significant bit. The bit having the greatest value.

I/O Abbreviation for Input and Output.

INDEX HOLE
The small hole in a floppy disk that is used by the computer for controlling where the data will be read or written.

INPUT Data that is entered into the computer from the keyboard or a peripheral.

INPUT/OUTPUT(I/O)
Refers to the reception (input) or transmission (output) of data to or from the computer from the peripherals. The hardware or software used for I/O is also included in what is called the I/O.

INSTRUCTION
A command in a program that the computer can execute. In a high-level language, an instruction may be several characters long.

INTERFACE Hardware and connections that do the I/O between different devices having different coding systems. For example, a printer interface takes data in the computer and converts it to printer data for output.

INTERPRETER
A program such as BASIC that interprets the commands as they are executed by the computer, as contrasted to a compiler or an assembler, in which the commands must first be placed in a machine language file.

INTERRUPT A system in which the computer is signalled when it is needed to process a program having higher priority. The advantage of using the interrupt system is that the computer does not need to poll for data, tying up computer time.

K A measure of memory or data file size. When K is capital, K = 1024, or 2 raised to the tenth power. When k is lower case, it is equal to 1000.

KEYWORD A word that is reserved by the programming language and therefore cannot be used in programs.

KILOBYTE 1,024bytes.

LANGUAGE Computer languages used on the Sanyo MBC-770 include SANYO BASIC, PASCAL, FORTH, and FORTRAN.

LOW-LEVEL LANGUAGE
Languages that are close to machine language, the language used by the computer.

LOW-ORDER BIT
The least significant bit. The bit having the smallest value.

MACHINE LANGUAGE
The language that the computer ultimately uses in executing a program. All programs written in high-level language must be converted to machine language before they can be executed by the computer.

MBC Sanyo abbreviation for Micro Business Computer. The MBC-series computers include the MBC-1100, MBC-1200, MBC-4000, MBC-550, and MBC-770 computers.

MEMORY LOCATION
A location in the memory that the computer can address. Each memory location has a unique address.

MEMORY MAP
A diagram or a list of the locations in memory that are allocated for system programs, ports, or user programs.

MICROCOMPUTER
A computer that uses a microprocessor for processing data.

MICROPROCESSOR
A single integrated circuit (IC) chip that performs I/O and executes machine language instructions.

MNEMONIC An abbreviation used to remember something that is more difficult to remember. In assembler, the commands are mnemonics.

MODE A state that the computer can enter in which a specified set of commands, rules, formats, or dates apply.

MODULO Sometimes abbreviated MOD, it is the remainder after division.

MONITOR
1) The CRT or TV that the computer uses for display.
2) The program that permits the user to examine and operate the computer on a byte level.

MS-DOS The disk operating system written by Microsoft to run on the 8088- and 8086- microprocessor based computers.

MULTIPLEXER
Hardware and software that takes a set of multiple inputs and creates one output.

NULL A string that does not contain characters, or a code or instruction that does nothing.

NIBBLE A half-byte, or four bits.

OBJECT CODE
The executable output file of an assembler or a compiler.

OFFSET A number added to an address to produce the address used when executing the program.

OPCODE Short for operation code. This is the number corresponding to the assembly language mnemonic.

OPERATING SYSTEM
The software program that controls the hardware and allows operator control of the various functions such as save, load, and copy.

OR A binary function whose output is "true" if at least one of its inputs is "true".

OUTPUT Data that is transferred from the computer to a peripheral.

PAGE
1) A screenful of information on a CRT display.
2) Data in a word processor that fills one sheet of paper.
3) A block of 256 bytes of memory.

PARAMETER
A value or character string that is substituted into the function upon execution.

PASCAL A high-level language named for Blaise Pascal that features structured programming.

PERIPHERAL
An external device that is attached to the computer. Peripherals are usually input or output devices.

PINOUT A list or diagram of the functions of each line on a connector or an integrated circuit chip.

PIXEL Short for picture element. A "dot" on the CRT display.

PORT An address in memory that is used by the computer for input/output operations.

PRINTED CIRCUIT (PC) BOARD
A board made of fiberglass or epoxy material that is etched with copper conductors on its surface. The components are soldered to the board.

PROCEDURE
A logically-separate, functional unit of a program.

PROGRAM A set of commands or instructions to the computer to accomplish a desired task.

PROM Acronym for "Programmable Read-Only Memory". PROMs are ROMs that can be reprogrammed, and are used in the same way as ROMs.

PROMPT A character string that the computer displays when it is ready for a new command.

QUEUE A set of programs awaiting processing. A wait list.

RANDOM ACCESS MEMORY (RAM)
The memory that can be read from and written to. The data in the RAM is volatile, it is lost when the power is turned off.

READ ONLY MEMORY (ROM)
Memory that can only be read. The data in the ROM is permanent, it is not lost when the computer is turned off.

RESIDENT A command or routine that is always present in main memory and thus does not have to be loaded each time it is called.

RETURN
1) The ENTER key on the keyboard.
2) A command in BASIC or a high-level language that is used at the end of a subroutine that tells the computer to complete the execution of the subroutine and return control to the main program.

ROUTINE A set of instructions in a program to perform a specific task.

RS-232C A standard interface and protocol for serial data communication between two devices.

SANYO The worldwide computer company that has brought you this computer series through its technical expertise.

SCAN LINE A single sweep of the electron beam across the screen or a CRT.

SCHEMATIC DIAGRAM
A diagram that shows in detail how the electronic components are interconnected.

SCROLL To cause the text on a display screen to move in the upward direction so as to either make room at the bottom for input of more data or to display what was below the text position.

SECTOR Part of a track of data on a floppy disk.

SEQUENTIAL FILE
A file which is organized so that the data is read and written in sequence, as opposed to a random-access file in which the data is accessed randomly.

SHELL The command interpreter for MS-DOS, a term adopted from the UNIX operating system.

SOFTWARE Programs, files, and data (intangible things) that are run on the computer.

SORT To alphabetize a list of words and numbers into increasing or decreasing order.

SOURCE The program written in the high-level language before compilation or assembly.

SPOOL Abbreviation for Simultaneous Peripheral Operations On Line. A file that will be output to a slow peripheral such as a printer is first output to a higher-speed peripheral such as a disk drive, and the printing is then done simultaneously while doing other operations to increase the system throughput.

STACK A reserved part of the memory that is used to store data temporarily on a last-in, first-out basis.

SUBROUTINE
An often-used portion of a program that can be called by the main program.

SYNTAX The set of rules and formats for a given language. In a computer, a spelling or other error in the input of a command may result in a "SYNTAX ERROR".

SYSTEM DISK
A special disk provided with the computer system that contains the operating system, languages, utility programs, monitors, and other software for system use. This disk is used at initialization.

TARGET The destination of the data or operation.

TASK The execution of a routine.

TEMPLATE A "form" of characters for later use with the function keys by the command processor.

TEXT Information entered for use by a program.

TOGGLE A two-position switch or function in an application program. The INS key toggles between insert OFF and ON.

TRACK A single concentric "band" where data is recorded on a floppy disk.

UTILITY A program to do general maintenance work such as copy files, transfer formats, sort, etc.

VARIABLE Numeric or alphabetic data that can be changed. A location in the memory where data is placed.

VECTOR In machine language, a set of instructions in table form that is used to redirect the activity of the microprocessor.

VERIFY A procedure in which once data is written to a floppy disk, the data is rechecked to check its reliability.

WARM START
Starting the computer by using a reset key or a reset instruction.

SANYO ELECTRIC CO.,LTD
Printed in Japan Sep. 1984
9376411922800 kw